

# Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

APPROX/RANDOM 2024, August 28–30, 2024,  
London School of Economics, London, UK

Edited by

Amit Kumar

Noga Ron-Zewi



*Editors*

**Amit Kumar** 

Indian Institute of Technology Delhi, New Delhi, India  
amit.kumar@cse.iitd.ac.in

**Noga Ron-Zewi** 

University of Haifa, Israel  
nronzewi@ds.haifa.ac.il

*ACM Classification 2012*

Theory of computation

**ISBN 978-3-95977-348-5**

*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/978-3-95977-348-5>.

*Publication date*

September, 2024

*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.2024.0

ISBN 978-3-95977-348-5

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 (Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University, Brno, CZ)
- Meena Mahajan (*Chair*, Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- 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)
- Pierre Senellart (ENS, Université PSL, Paris, FR)

**ISSN 1868-8969**

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



## ■ Contents

|   |        |
|---|--------|
| Preface                                   |        |
| <i>Amit Kumar and Noga Ron-Zewi</i> ..... | 0:xi   |
| Program Committees                        |        |
| .....                                     | 0:xiii |
| Subreviewers                              |        |
| .....                                     | 0:xv   |
| List of Authors                           |        |
| .....                                     | 0:xix  |

## APPROX

|  |            |
|--|------------|
| A $(3/2 + 1/e)$ -Approximation Algorithm for Ordered TSP   |            |
| <i>Susanne Armbruster, Matthias Mnich, and Martin Nägele</i> .....                                 | 1:1–1:18   |
| Online Time-Windows TSP with Predictions   |            |
| <i>Shuchi Chawla and Dimitris Christou</i> .....   | 2:1–2:21   |
| Degrees and Network Design: New Problems and Approximations  |            |
| <i>Michael Dinitz, Guy Kortsarz, and Shi Li</i> .....  | 3:1–3:17   |
| Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center                                       |            |
| <i>Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Saket Saurabh, and Meirav Zehavi</i> .....    | 4:1–4:19   |
| Asynchronous Majority Dynamics on Binomial Random Graphs   |            |
| <i>Divyarthi Mohan and Pawel Pralat</i> .....  | 5:1–5:20   |
| Bipartizing (Pseudo-)Disk Graphs: Approximation with a Ratio Better than 3                         |            |
| <i>Daniel Lokshтанov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi</i> ..              | 6:1–6:14   |
| A Logarithmic Approximation of Linearly-Ordered Colourings   |            |
| <i>Johan Håstad, Björn Martinsson, Tamio-Vesa Nakajima, and Stanislav Živný</i> ....               | 7:1–7:6    |
| Speed-Robust Scheduling Revisited  |            |
| <i>Josef Minařík and Jiří Sgall</i> .....  | 8:1–8:20   |
| On the Generalized Mean Densest Subgraph Problem: Complexity and Algorithms                        |            |
| <i>Karthekeyan Chandrasekaran, Chandra Chekuri, Manuel R. Torres, and Weihao Zhu</i> .....         | 9:1–9:21   |
| Improved Online Load Balancing with Known Makespan   |            |
| <i>Martin Böhm, Matej Lieskovský, Sören Schmitt, Jiří Sgall, and Rob van Stee</i> ....             | 10:1–10:21 |
| On the NP-Hardness Approximation Curve for Max-2Lin(2)   |            |
| <i>Björn Martinsson</i> .....  | 11:1–11:38 |
| Universal Optimization for Non-Clairvoyant Subadditive Joint Replenishment                         |            |
| <i>Tomer Ezra, Stefano Leonardi, Michał Pawłowski, Matteo Russo, and Seeun William Umboh</i> ..... | 12:1–12:24 |

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi



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

|   |            |
|---|------------|
| The Average-Value Allocation Problem<br><i>Kshipra Bhawalkar, Zhe Feng, Anupam Gupta, Aranyak Mehta, David Wajc,<br/>and Di Wang</i> .....  | 13:1–13:23 |
| Scheduling on a Stochastic Number of Machines<br><i>Moritz Buchem, Franziska Eberle, Hugo Kooki Kasuya Rosado, Kevin Schewior,<br/>and Andreas Wiese</i> .....  | 14:1–14:15 |
| Distributional Online Weighted Paging with Limited Horizon<br><i>Yaron Fairstein, Joseph (Seffi) Naor, and Tomer Tsachor</i> .....  | 15:1–15:15 |
| Weighted Matching in the Random-Order Streaming and Robust Communication<br>Models<br><i>Diba Hashemi and Weronika Wrzos-Kaminska</i> .....   | 16:1–16:26 |
| Competitive Query Minimization for Stable Matching with One-Sided Uncertainty<br><i>Evrpidis Bampis, Konstantinos Dogeas, Thomas Erlebach, Nicole Megow,<br/>Jens Schölter, and Amitabh Trehan</i> .....    | 17:1–17:21 |
| A Constant Factor Approximation for Directed Feedback Vertex Set in Graphs<br>of Bounded Genus<br><i>Hao Sun</i> .....  | 18:1–18:20 |
| More Basis Reduction for Linear Codes: Backward Reduction, BKZ, Slide<br>Reduction, and More<br><i>Surendra Ghentiyala and Noah Stephens-Davidowitz</i> .....   | 19:1–19:22 |
| Online $k$ -Median with Consistent Clusters<br><i>Benjamin Moseley, Heather Newman, and Kirk Pruhs</i> .....  | 20:1–20:22 |
| The Telephone $k$ -Multicast Problem<br><i>Daniel Hathcock, Guy Kortsarz, and R. Ravi</i> .....   | 21:1–21:16 |
| Scheduling Splittable Jobs on Configurable Machines<br><i>Matthew Casey, Rajmohan Rajaraman, David Stalfa, and Cheng Tan</i> .....  | 22:1–22:20 |
| On Instance-Optimal Algorithms for a Generalization of Nuts and Bolts and<br>Generalized Sorting<br><i>Mayank Goswami and Riko Jacob</i> .....  | 23:1–23:23 |
| Learning-Augmented Maximum Independent Set<br><i>Vladimir Braverman, Prathamesh Dharangutte, Vihan Shah, and Chen Wang</i> ....   | 24:1–24:18 |
| Maximum Unique Coverage on Streams: Improved FPT Approximation Scheme<br>and Tighter Space Lower Bound<br><i>Philip Cervenjak, Junhao Gan, Seun William Umboh, and Anthony Wirth</i> .....                  | 25:1–25:23 |
| Improved Streaming Algorithm for the Klee’s Measure Problem and<br>Generalizations<br><i>Mridul Nandi, N. V. Vinodchandran, Arijit Ghosh, Kuldeep S. Meel,<br/>Soumit Pal, and Sourav Chakraborty</i> ..... | 26:1–26:20 |
| An EPTAS for Cardinality Constrained Multiple Knapsack via Iterative<br>Randomized Rounding<br><i>Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai</i> .....  | 27:1–27:17 |

Rectangle Tiling Binary Arrays  
*Pratik Ghosal, Syed Mohammad Meesum, and Katarzyna Paluch* ..... 28:1–28:17

Approximation Algorithms for Correlated Knapsack Orienteering  
*David Alemán Espinosa and Chaitanya Swamy* ..... 29:1–29:24

Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem  
*Gabriel Arpino, Daniil Dmitriev, and Nicolo Grometto* ..... 30:1–30:22

**RANDOM**

The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced  
*Amnon Ta-Shma and Ron Zadichario* ..... 31:1–31:22

Near-Linear Time Samplers for Matroid Independent Sets with Applications  
*Xiaoyu Chen, Heng Guo, Xinyuan Zhang, and Zongrui Zou* ..... 32:1–32:12

On the Amortized Complexity of Approximate Counting  
*Ishaq Aden-Ali, Yanjun Han, Jelani Nelson, and Huacheng Yu* ..... 33:1–33:17

Matrix Multiplication Reductions  
*Ashish Gola, Igor Shinkar, and Harsimran Singh* ..... 34:1–34:15

Testing Intersectingness of Uniform Families  
*Ishay Haviv and Michal Parnas* ..... 35:1–35:15

On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs  
*Lap Chi Lau and Dante Tjowasi* ..... 36:1–36:18

Nearly Optimal Bounds for Sample-Based Testing and Learning of  $k$ -Monotone Functions  
*Hadley Black* ..... 37:1–37:23

Approximating the Number of Relevant Variables in a Parity Implies Proper Learning  
*Nader H. Bshouty and George Haddad* ..... 38:1–38:15

The Number of Random 2-SAT Solutions Is Asymptotically Log-Normal  
*Arnab Chatterjee, Amin Coja-Oghlan, Noela Müller, Connor Riddlesden, Maurice Rolvien, Pavel Zakharov, and Haodong Zhu* ..... 39:1–39:15

Private Counting of Distinct Elements in the Turnstile Model and Extensions  
*Monika Henzinger, A. R. Sricharan, and Teresa Anna Steiner* ..... 40:1–40:21

Hilbert Functions and Low-Degree Randomness Extractors  
*Alexander Golovnev, Zeyu Guo, Pooya Hatami, Satyajeet Nagargoje, and Chao Yan* ..... 41:1–41:24

Matrix Multiplication Verification Using Coding Theory  
*Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn Warton* ..... 42:1–42:13

Interactive Coding with Unbounded Noise  
*Eden Fargion, Ran Gelles, and Meghal Gupta* ..... 43:1–43:16

Optimal Pseudorandom Generators for Low-Degree Polynomials over Moderately Large Fields  
*Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk* ..... 44:1–44:19

|   |            |
|---|------------|
| Refining the Adaptivity Notion in the Huge Object Model<br><i>Tomer Adar and Eldar Fischer</i> .....  | 45:1–45:16 |
| Support Testing in the Huge Object Model<br><i>Tomer Adar, Eldar Fischer, and Amit Levi</i> .....   | 46:1–46:16 |
| Upper Bounds on the 2-Colorability Threshold of Random $d$ -Regular $k$ -Uniform Hypergraphs for $k \geq 3$<br><i>Evan Chang, Neel Kolhe, and Youngtak Sohn</i> ..... | 47:1–47:23 |
| Improved Bounds for High-Dimensional Equivalence and Product Testing Using Subcube Queries<br><i>Tomer Adar, Eldar Fischer, and Amit Levi</i> .....                   | 48:1–48:21 |
| Parallelising Glauber Dynamics<br><i>Holden Lee</i> .....   | 49:1–49:24 |
| Towards Simpler Sorting Networks and Monotone Circuits for Majority<br><i>Natalia Dobrokhotova-Maikova, Alexander Kozachinskiy, and Vladimir Podolskii</i> .....      | 50:1–50:18 |
| Consequences of Randomized Reductions from SAT to Time-Bounded Kolmogorov Complexity<br><i>Halley Goldberg and Valentine Kabanets</i> .....                           | 51:1–51:19 |
| Trace Reconstruction from Local Statistical Queries<br><i>Xi Chen, Anindya De, Chin Ho Lee, and Rocco A. Servedio</i> .....   | 52:1–52:24 |
| When Do Low-Rate Concatenated Codes Approach The Gilbert–Varshamov Bound?<br><i>Dean Doron, Jonathan Mosheiff, and Mary Wootters</i> .....                            | 53:1–53:12 |
| Parallel Repetition of $k$ -Player Projection Games<br><i>Amey Bhangale, Mark Braverman, Subhash Khot, Yang P. Liu, and Dor Minzer</i> .....                          | 54:1–54:16 |
| Faster Algorithms for Schatten- $p$ Low Rank Approximation<br><i>Praneeth Kacham and David P. Woodruff</i> .....  | 55:1–55:19 |
| Fast and Slow Mixing of the Kawasaki Dynamics on Bounded-Degree Graphs<br><i>Aiya Kuchukova, Marcus Pappik, Will Perkins, and Corrine Yap</i> .....                   | 56:1–56:24 |
| Stochastic Distance in Property Testing<br><i>Uri Meir, Gregory Schwartzman, and Yuichi Yoshida</i> .....   | 57:1–57:13 |
| Expanderizing Higher Order Random Walks<br><i>Vedat Levi Alev and Shravas Rao</i> .....   | 58:1–58:24 |
| Ramsey Properties of Randomly Perturbed Hypergraphs<br><i>Elad Aigner-Horev, Dan Hefetz, and Mathias Schacht</i> .....  | 59:1–59:18 |
| Nearly Optimal Local Algorithms for Constructing Sparse Spanners of Clusterable Graphs<br><i>Reut Levi, Moti Medina, and Omer Tubul</i> .....                         | 60:1–60:21 |
| When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?<br><i>Kuan Cheng, Minghui Ouyang, Chong Shangguan, and Yuanting Shen</i> .....                   | 61:1–61:23 |



|  |            |
|--|------------|
| Coboundary and Cosystolic Expansion Without Dependence on Dimension or Degree<br><i>Yotam Dikstein and Irit Dinur</i> .....  | 62:1–62:24 |
| Rapid Mixing of the Down-Up Walk on Matchings of a Fixed Size<br><i>Vishesh Jain and Clayton Mizgerd</i> .....   | 63:1–63:13 |
| On the Communication Complexity of Finding a King in a Tournament<br><i>Nikhil S. Mande, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh</i> .....            | 64:1–64:23 |
| Capacity-Achieving Gray Codes<br><i>Venkatesan Guruswami and Hsin-Po Wang</i> .....  | 65:1–65:9  |
| On Black-Box Meta Complexity and Function Inversion<br><i>Noam Mazon and Rafael Pass</i> .....   | 66:1–66:12 |
| Explicit and Near-Optimal Construction of $t$ -Rankwise Independent Permutations<br><i>Nicholas Harvey and Arvin Sahami</i> .....                                  | 67:1–67:12 |
| Sparse High Dimensional Expanders via Local Lifts<br><i>Inbar Ben Yaacov, Yotam Dikstein, and Gal Maor</i> .....   | 68:1–68:24 |
| Randomness Extractors in $AC^0$ and $NC^1$ : Optimal up to Constant Factors<br><i>Kuan Cheng and Ruiyang Wu</i> .....  | 69:1–69:22 |
| On Sampling from Ising Models with Spectral Constraints<br><i>Andreas Galanis, Alkis Kalavasis, and Anthimos Vardis Kandiros</i> .....                             | 70:1–70:14 |
| Approximate Degree Composition for Recursive Functions<br><i>Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, and Nitin Saurabh</i> .....     | 71:1–71:17 |
| Public Coin Interactive Proofs for Label-Invariant Distribution Properties<br><i>Tal Herman</i> .....  | 72:1–72:23 |
| Additive Noise Mechanisms for Making Randomized Approximation Algorithms Differentially Private<br><i>Jakub Tětek</i> .....  | 73:1–73:20 |
| Improved Bounds for Graph Distances in Scale Free Percolation and Related Models<br><i>Kostas Lakis, Johannes Lengler, Kalina Petrova, and Leon Schiller</i> ..... | 74:1–74:22 |
| Derandomizing Multivariate Polynomial Factoring for Low Degree Factors<br><i>Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf</i> .....                          | 75:1–75:20 |



## ■ Preface

This volume contains the papers presented at the 27th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2024) and the 28th International Conference on Randomization and Computation (RANDOM 2024). APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally-difficult problems, and the 2024 edition was the 27th in the series. RANDOM is concerned with applications of randomness to computational and combinatorial problems, and the 2024 edition was the 28th in the series. The two conferences were held in parallel at the London School of Economics and Political Science (LSE), London, UK during August 28–30, 2024. This year, the plenary speaker for APPROX was Anupam Gupta from New-York University, and the plenary speaker for RANDOM was Mark Jerrum from Queen Mary University of London.

Topics of interest for APPROX include 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. Those at RANDOM include the 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, sublinear-time and local algorithms, computational learning theory, and the role of (pseudo)randomness in other areas of computer science such as cryptography, data privacy, and quantum information.

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 Atlanta (2023). In 2020, 2021, and 2022, the conferences were held online.

The volume contains 30 contributed papers selected by the APPROX Program Committee out of 58 submissions, and 45 contributed papers selected by the RANDOM Program Committee out of 95 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, Nicole Megow, Laura Sanità, Chaitanya Swamy, 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. We would also like to thank the local organizing committee at LSE Ahmad Abdi, Tugkan Batu, Neil Olver, and Gergory Sorokin, and the Dagstuhl Publishing Team for their dedicated handling of the proceedings.

It was our pleasure to serve as chairs of the Approx/Random 2024 Program Committees, and to edit this volume of the proceedings.

Amit Kumar and Noga Ron-Zewi

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi



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



## ■ Program Committees

### APPROX

|                  |  |
|------------------|--|
| Susanne Albers   | Technical University of Munich           |
| Keerti Choudhary | Indian Institute of Technology Delhi     |
| Debarati Das     | Pennsylvania State University            |
| Thomas Erlebach  | Durham University                        |
| Zhiyi Huang      | University of Hong Kong                  |
| Sanjeev Khanna   | University of Pennsylvania               |
| Amit Kumar       | Indian Institute of Technology Delhi     |
| Euiwoong Lee     | University of Michigan                   |
| Roie Levin       | Rutgers University                       |
| Anand Louis      | Indian Institute of Science Bangalore    |
| Julián Mestre    | University of Sydney                     |
| Sahil Singla     | Georgia Institute of Technology          |
| Thomas Rothvoss  | University of Washington                 |
| José Verschae    | Pontificia Universidad Católica de Chile |
| Erik Waingarten  | University of Pennsylvania               |
| David Wajc       | Technion–Israel Institute of Technology  |

### RANDOM

|                          |   |
|--------------------------|---|
| Ioana Bercea             | KTH Royal Institute of Technology             |
| Andrej Bogdanov          | University of Ottawa                          |
| Sarah Cannon             | Claremont McKenna College                     |
| Lijie Chen               | UC Berkeley                                   |
| Weiming Feng             | ETH Zurich                                    |
| Sumegha Garg             | Rutgers University                            |
| Zeyu Guo                 | Ohio State University                         |
| Tom Gur                  | University of Cambridge                       |
| William Hoza             | University of Chicago                         |
| Valentine Kabanets       | Simon Fraser University                       |
| Reut Levi                | Reichman University                           |
| Ray Li                   | Santa Clara University                        |
| Inbal Livni-Navon        | Stanford University                           |
| Dor Minzer               | MIT   |
| Ilan Newman              | University of Haifa                           |
| Pan Peng                 | University of Science and Technology of China |
| João Ribeiro             | Universidade Nova de Lisboa                   |
| Noga Ron-Zewi (PC Chair) | University of Haifa                           |
| Piyush Srivastava        | Tata Institute of Fundamental Research        |
| Emanuele Viola           | Northeastern University                       |
| Erik Waingarten          | University of Pennsylvania                    |
| Ronald de Wolf           | CWI and University of Amsterdam               |





## ■ Subreviewers

### APPROX

|                          |                         |
|--------------------------|-------------------------|
| Aditya Anand             | Haris Aziz              |
| Aritra Banik             | Libor Barto             |
| Anubhav Baweja           | Soheil Behnezhad        |
| Petra Berenbrink         | Anup Bhattacharya       |
| Hans-Joachim Böckenhauer | Sander Borst            |
| Katrin Casel             | Diptarka Chakraborty    |
| Rajesh Chitnis           | Christian Coester       |
| Minati De                | Christoph Dürr          |
| Franziska Eberle         | Yuri Faenza             |
| Guichen Gao              | Prantar Ghosh           |
| Suprovat Ghoshal         | Rohan Ghuge             |
| Jacob Gilbert            | Sander Gribling         |
| Guru Guruganesh          | Ruben Hoeksma           |
| Lunjia Hu                | Neng Huang              |
| Tanmay Inamdar           | Peyman Jabbarzade       |
| Rhea Jain                | Klaus Jansen            |
| Rajesh Jayaram           | Rajesh Jayaram          |
| Shaofeng H.-C. Jiang     | John Kallaughner        |
| Arindam Khan             | Arindam Khan            |
| Evangelos Kipouridis     | Kim-Manuel Klein        |
| Yusuke Kobayashi         | Christian Konrad        |
| Venkata Koppula          | Rajendra Kumar          |
| Tomer Lange              | Thomas Lavastida        |
| Shi Li                   | Alexander Lindermayr    |
| Andreas Maggiori         | Andrew McGregor         |
| Arturo Merino            | Pranabendu Misra        |
| Seffi Naor               | Meghana Nasre           |
| Prajakta Nimbhorkar      | Kalen Patton            |
| Rameesh Paul             | Marcin Pilipczuk        |
| Madhusudhan Reddy Pittu  | Kirk Pruhs              |
| Aaron Putterman          | Saladi Rahul            |
| Malin Rau                | Arka Ray                |
| Mirabel Reid             | Raghuvansh Saxena       |
| Kevin Schewior           | Niklas Schlomberg       |
| Amatya Sharma            | Sandeep Silwal          |
| Apoorv Vikram Singh      | Enze Sun                |
| Janani Sundaresan        | Vaishali Surianarayanan |
| Varun Suriyanarayana     | Noam Touitou            |
| Victor Verdugo           | Pavel Veselý            |
| Hao-Ting Wei             | Chenyang Xu             |
| Zhiyang Xun              | Ilias Zadik             |
| Ahad N. Zehmakan         | Qiankun Zhang           |
| Ruilong Zhang            | Yuhao Zhang             |
| Mingxian Zhong           |                         |

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi



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

**RANDOM**

|                        |                          |
|------------------------|--------------------------|
| Maryam Aliakbarpour    | Hugo Aaronson            |
| Noga Alon              | Omar Alrabiah            |
| Matthew Anderson       | Robert Andrews           |
| Paul Bastide           | Ivona Bezakova           |
| Siddharth Bhandari     | Amey Bhangale            |
| Eric Blais             | Jaroslav Blasiok         |
| Avrim Blum             | Greg Bodwin              |
| Jan van den Brand      | Vladimir Braverman       |
| Karl Bringmann         | Nader Bshouty            |
| Thach V. Bui           | Mark Bun                 |
| Clement Canonne        | Charlie Carlson          |
| Anamay Chaturvedi      | Zongchen Chen            |
| Mahdi Cheraghchi       | Tsun Ming Cheung         |
| Gil Cohen              | Amin Coja-Oghlan         |
| Graham Cormode         | Varsha Dani              |
| Shagnik Das            | Chengyuan Deng           |
| Prathamesh Dharangutte | Yotam Dikstein           |
| Michal Dori            | Dean Doron               |
| Pranjal Dutta          | Charilaos Eftymiou       |
| Asaf Ferber            | Eldar Fischer            |
| Nick Fischer           | Andreas Galanis          |
| Margalit Glasgow       | Andreas Göbel            |
| Oded Goldreich         | Alexander Golovnev       |
| Louis Golowich         | Subhajit Goswami         |
| Peter Gracar           | Catherine Greenhill      |
| Hermann Gruber         | Nathaniel Harms          |
| Pooya Hatami           | Tal Herman               |
| Markus Heydenreich     | Shuichi Hirahara         |
| Max Hopkins            | Kaave Hosseini           |
| Mark Huber             | Rahul Ilango             |
| Vishesh Jain           | Fernando Granha Jeronimo |
| Ce Jin                 | Ralph Keusch             |
| Swastik Kopparty       | Evangelos Kosinas        |
| Akash Kumar            | Vinayak Kumar            |
| Dmitriy Kunisky        | Orna Kupferman           |
| Matthew Kwan           | Jane Lange               |
| Christian Janos Lebeda | Chin Ho Lee              |
| Troy Lee               | Anthony Leverrier        |
| Amit Levi              | Vedat Levi Alev          |
| Yi Li                  | Jyun-Jie Liao            |
| Hongyang Liu           | Kuikui Liu               |
| Peihan Liu             | Yang Liu                 |
| Yanyi Liu              | David Rasmussen Lolck    |
| Zhenjian Lu            | Xin Lyu                  |
| Peter Manohar          | Samuel McCauley          |
| Moti Medina            | Parth Mittal             |
| Sidhanth Mohanty       | Cristopher Moore         |



|                     |                         |
|---------------------|-------------------------|
| Ben Morris          | Patrick Morris          |
| Tamalika Mukherjee  | Shivam Nadimpalli       |
| Ofer Neiman         | Sam Olesker-Taylor      |
| Neil Olver          | Konstantinos Panagiotou |
| Pedro Paredes       | Phevos Paschalidis      |
| Shyamal Patel       | Richard Peng            |
| Will Perkins        | Huy Tuan Pham           |
| Diogo Poças         | Aditya Potukuchi        |
| Kršjanis Prusis     | Aaron Putterman         |
| Edward Pyne         | Ninad Rajgopal          |
| C Ramya             | Shravas Rao             |
| Hanlin Ren          | Ramprasad Saptharishi   |
| Raghuvansh Saxena   | Jonathan Scarlett       |
| Rocco Servedio      | Chong Shangguan         |
| Asaf Shapira        | Kshiteej Sheth          |
| Nobutaka Shimizu    | Morgan Shirley          |
| Mihir Singhal       | Makrand Sinha           |
| Adam Smith          | Christian Sohler        |
| Adarsh Srinivasan   | Srikanth Srinivasan     |
| Teresa Anna Steiner | Uri Stemmer             |
| Madhu Sudan         | He Sun                  |
| Janani Sundaresan   | Xinyu Tan               |
| Roei Tell           | Anamay Tengse           |
| Salil Vadhan        | Eric Vigoda             |
| Ben Lee Volk        | Thuy Duong Vuong        |
| Chen Wang           | Dingyu Wang             |
| Hsin-Po Wang        | Jiaheng Wang            |
| Yuval Wigderson     | Karl Wimmer             |
| Hongxun Wu          | Qiping Yang             |
| Yuichi Yoshida      | Chen Yuan               |
| Ahad N. Zehmakan    | Zihan Zhang             |
| Kai Zhe Zheng       | Hong Zhou               |
| Samson Zhou         |                         |



## ■ List of Authors

- Tomer Adar  (45, 46, 48)  
Technion – Israel Institute of Technology,  
Haifa, Israel
- Ishaq Aden-Ali  (33)  
University of California, Berkeley, CA, USA
- Elad Aigner-Horev  (59)  
School of Computer Science,  
Ariel University, Israel
- David Alemán Espinosa (29)  
Dept. of Combinatorics and Optimization, Univ.  
Waterloo, Waterloo, ON N2L 3G1, Canada
- Vedat Levi Alev  (58)  
Hebrew University of Jerusalem, Israel
- Susanne Armbruster  (1)  
Research Institute for Discrete Mathematics and  
Hausdorff Center for Mathematics, University of  
Bonn, Germany
- Gabriel Arpino  (30)  
University of Cambridge, UK
- Evipidis Bampis  (17)  
Sorbonne Université, CNRS, LIP6, Paris, France
- Inbar Ben Yaacov  (68)  
Weizmann Institute of Science, Rehovot, Israel
- Huck Bennett  (42)  
University of Colorado Boulder, CO, USA
- Amey Bhangale  (54)  
Department of Computer Science and  
Engineering, University of California, Riverside,  
CA, USA
- Kshipra Bhawalkar  (13)  
Google Research, Mountain View, USA
- Hadley Black  (37)  
University of California, San Diego, USA
- Mark Braverman  (54)  
Department of Computer Science, Princeton  
University, NJ, USA
- Vladimir Braverman (24)  
Rice University, Houston, TX, USA;  
Google Research
- Nader H. Bshouty  (38)  
Department of Computer Science,  
Technion, Israel
- Moritz Buchem  (14)  
Technische Universität München, Germany
- Martin Böhm  (10)  
University of Wrocław, Poland
- Matthew Casey  (22)  
Northeastern University, Boston MA 02115,  
USA
- Philip Cervenjak  (25)  
School of Computing and Information Systems,  
The University of Melbourne, Australia
- Sourav Chakraborty (26, 71)  
Indian Statistical Institute, Kolkata, India
- Karthekeyan Chandrasekaran  (9)  
University of Illinois, Urbana-Champaign, USA
- Evan Chang (47)  
Massachusetts Institute of Technology, USA
- Arnab Chatterjee (39)  
Department of Computer Science, TU  
Dortmund, Germany
- Shuchi Chawla  (2)  
University of Texas at Austin, United States
- Chandra Chekuri  (9)  
University of Illinois, Urbana-Champaign, USA
- Xi Chen  (52)  
Columbia University, New York, NY, USA
- Xiaoyu Chen (32)  
State Key Laboratory for Novel Software  
Technology, New Cornerstone Science  
Laboratory, Nanjing University, 163 Xianlin  
Avenue, Nanjing, Jiangsu Province, China
- Kuan Cheng  (61, 69)  
Center on Frontiers of Computing Studies,  
School of Computer Science, Peking University,  
China
- Dimitris Christou  (2)  
University of Texas at Austin, United States
- Amin Coja-Oghlan (39)  
Department of Computer Science,  
TU Dortmund, Germany
- Anindya De  (52)  
University of Pennsylvania,  
Philadelphia, PA, USA

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi



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

- Prathamesh Dharangutte (24)  
Rutgers University, NJ, USA
- Yotam Dikstein  (62, 68)  
Institute for Advanced Study,  
Princeton, NJ, USA
- Michael Dinitz (3)  
Johns Hopkins University, Baltimore, MD, USA
- Irit Dinur  (62)  
Weizmann Institute of Science, Rehovot, Israel
- Daniil Dmitriev  (30)  
ETH Zürich and ETH AI Center, Switzerland
- Natalia Dobrokhotova-Maikova  (50)  
Yandex, Tel Aviv, Israel
- Konstantinos Dogeas  (17)  
Department of Computer Science, Durham  
University, Durham, United Kingdom
- Dean Doron  (53)  
Ben-Gurion University of the Negev,  
Beersheba, Israel
- Ilan Doron-Arad  (27)  
Computer Science Department,  
Technion, Haifa, Israel
- Pranjali Dutta  (75)  
School of Computing, National University of  
Singapore, Singapore
- Ashish Dwivedi  (44)  
Department of Computer Science and  
Engineering, The Ohio State University,  
Columbus, OH, USA
- Franziska Eberle  (14)  
Technische Universität Berlin, Germany
- Thomas Erlebach  (17)  
Department of Computer Science, Durham  
University, Durham, United Kingdom
- Tomer Ezra (12)  
Harvard University, Cambridge, MA, USA
- Yaron Fairstein  (15)  
Amazon.com, Haifa, Israel
- Eden Fargion  (43)  
Faculty of Engineering, Bar-Ilan University,  
Ramat-Gan, Israel
- Zhe Feng  (13)  
Google Research, Mountain View, USA
- Eldar Fischer (45, 46, 48)  
Technion – Israel Institute of Technology,  
Haifa, Israel
- Fedor V. Fomin  (4)  
University of Bergen, Norway
- Karthik Gajulapalli  (42)  
Georgetown University, Washington DC, USA
- Andreas Galanis (70)  
University of Oxford, UK
- Junhao Gan  (25)  
School of Computing and Information Systems,  
The University of Melbourne, Australia
- Ran Gelles  (43)  
Faculty of Engineering, Bar-Ilan University,  
Ramat-Gan, Israel
- Surendra Ghentiyala (19)  
Cornell University, Ithaca, NY, USA
- Pratik Ghosal  (28)  
Indian Institute of Technology, Palakkad, India
- Arijit Ghosh (26)  
Indian Statistical Institute, Kolkata, India
- Ashish Gola (34)  
Simon Fraser University, Burnaby, Canada
- Halley Goldberg (51)  
Simon Fraser University, Burnaby, Canada
- Petr A. Golovach  (4)  
University of Bergen, Norway
- Alexander Golovnev  (41, 42)  
Georgetown University, Washington, DC, USA
- Mayank Goswami  (23)  
Queens College CUNY, Flushing, New York,  
USA
- Nicolo Grometto (30)  
Princeton University, USA
- Heng Guo (32)  
School of Informatics, University of Edinburgh,  
Informatics Forum, 10 Crichton Street,  
Edinburgh, EH8 9AB, UK
- Zeyu Guo  (41, 44)  
The Ohio State University, Columbus, OH,  
United States of America
- Anupam Gupta  (13)  
NYU & Google Research, New York City &  
Mountain View, USA
- Meghal Gupta  (43)  
University of California, Berkeley, CA, USA
- Venkatesan Guruswami  (65)  
University of California, Berkeley, CA, USA


- George Haddad (38)  
Department of Computer Science,  
Technion, Israel
- YanJun Han  (33)  
New York University, NY, USA
- Nicholas Harvey  (67)  
Department of Computer Science and  
Department of Mathematics, University of  
British Columbia, Vancouver, Canada
- Diba Hashemi  (16)  
EPFL, Lausanne, Switzerland
- Pooya Hatami  (41)  
The Ohio State University, Columbus, OH,  
United States of America
- Daniel Hathcock  (21)  
Carnegie Mellon University, United States
- Ishay Haviv (35)  
The Academic College of Tel Aviv-Yaffo,  
Tel Aviv 61083, Israel
- Dan Hefetz  (59)  
School of Computer Science,  
Ariel University, Israel
- Monika Henzinger  (40)  
Institute of Science and Technology,  
Klosterneuburg, Austria
- Tal Herman  (72)  
Weizmann Institute of Science, Rehovot, Israel
- Johan Håstad  (7)  
KTH Royal Institute of Technology,  
Stockholm, Sweden
- Tanmay Inamdar  (4)  
Indian Institute of Technology Jodhpur,  
Jodhpur, India
- Riko Jacob  (23)  
IT University of Copenhagen,  
København S, Denmark
- Vishesh Jain  (63)  
Department of Mathematics, Statistics, and  
Computer Science, University of Illinois Chicago,  
Chicago, IL, USA
- Valentine Kabanets (51)  
Simon Fraser University, Burnaby, Canada
- Praneeth Kacham  (55)  
Carnegie Mellon University,  
Pittsburgh, PA, USA;  
Google Research, New York, USA
- Alkis Kalavasis (70)  
Yale University, New Haven, CT, USA
- Anthimos Vardis Kandiros (70)  
Massachusetts Institute of Technology,  
Cambridge, MA, USA
- Hugo Kooki Kasuya Rosado  (14)  
Technische Universität München, Germany
- Chandrima Kayal (71)  
Indian Statistical Institute, Kolkata, India
- Subhash Khot  (54)  
Department of Computer Science, Courant  
Institute of Mathematical Sciences, New York  
University, NY, USA
- Neel Kolhe (47)  
University of California, Berkeley, USA
- Guy Kortsarz (3, 21)  
Rutgers University, Camden, NJ, USA
- Alexander Kozachinskiy  (50)  
IMFD & CENIA, Santiago, Chile
- Aiya Kuchukova  (56)  
School of Mathematics, Georgia Institute of  
Technology, Atlanta, GA, USA
- Ariel Kulik  (27)  
Computer Science Department,  
Technion, Haifa, Israel
- Kostas Lakis  (74)  
ETH Zürich, Department of Computer Science,  
Zürich, Switzerland
- Lap Chi Lau (36)  
Cheriton School of Computer Science,  
University of Waterloo, Canada
- Chin Ho Lee  (52)  
North Carolina State University,  
Raleigh, NC, USA
- Holden Lee  (49)  
Department of Applied Mathematics and  
Statistics, The Johns Hopkins University,  
Baltimore, MD, USA
- Johannes Lengler (74)  
ETH Zürich, Department of Computer Science,  
Zürich, Switzerland
- Stefano Leonardi (12)  
Sapienza University of Rome, Italy
- Amit Levi  (46, 48)  
University of Haifa, Israel


- Reut Levi  (60)  
Efi Arazi School of Computer Science,  
Reichman University, Herzliya, Israel
- Shi Li (3)  
Nanjing University, Jiangsu, China
- Matej Lieskovský  (10)  
Computer Science Institute of Charles  
University, Faculty of Mathematics and Physics,  
Prague, Czechia
- Yang P. Liu  (54)  
School of Mathematics, Institute for Advanced  
Study, Princeton, NJ, USA
- Daniel Lokshtanov  (6)  
University of California, Santa Barbara, USA
- Nikhil S. Mande  (64)  
University of Liverpool, UK
- Gal Maor  (68)  
Tel Aviv University, Tel Aviv, Israel
- Björn Martinsson  (7, 11)  
KTH Royal Institute of Technology,  
Stockholm, Sweden
- Noam Mazon  (66)  
Tel Aviv University, Israel
- Moti Medina  (60)  
Faculty of Engineering, Bar-Ilan University,  
Ramat Gan, Israel
- Kuldeep S. Meel (26)  
University of Toronto, Canada
- Syed Mohammad Meesum  (28)  
Krea University, India
- Nicole Megow  (17)  
Faculty of Mathematics and Computer Science,  
University of Bremen, Bremen, Germany
- Aranyak Mehta  (13)  
Google Research, Mountain View, USA
- Uri Meir  (57)  
Blavatnik School of Computer Science,  
Tel Aviv University, Tel Aviv, Israel
- Josef Minařík (8)  
Computer Science Institute of Charles Univ.,  
Faculty of Mathematics and Physics, Prague,  
Czechia
- Dor Minzer  (54)  
Department of Mathematics, Massachusetts  
Institute of Technology, Cambridge, MA, USA
- Rajat Mittal (71)  
Indian Institute of Technology Kanpur, India
- Clayton Mizgerd (63)  
Department of Mathematics, Statistics, and  
Computer Science, University of Illinois Chicago,  
Chicago, IL, USA
- Matthias Mnich  (1)  
Hamburg University of Technology, Institute for  
Algorithms and Complexity, Hamburg, Germany
- Divyarthi Mohan  (5)  
Blavatnik School of Computer Science,  
Tel Aviv University, Israel
- Benjamin Moseley  (20)  
Tepper School of Business, Carnegie Mellon  
University, Pittsburgh, PA, USA
- Jonathan Mosheiff  (53)  
Ben-Gurion University of the Negev,  
Beersheba, Israel
- Noela Müller (39)  
Department of Mathematics and Computer  
Science, Eindhoven University of Technology,  
Netherlands
- Satyajeet Nagargoje  (41)  
Georgetown University, Washington, DC,  
United States of America
- Tamio-Vesa Nakajima  (7)  
Department of Computer Science,  
University of Oxford, UK
- Mridul Nandi (26)  
Indian Statistical Institute, Kolkata, India
- Joseph (Seffi) Naor (15)  
Computer Science Department,  
Technion, Haifa, Israel
- Jelani Nelson  (33)  
University of California, Berkeley, CA, USA
- Heather Newman  (20)  
Department of Mathematical Sciences, Carnegie  
Mellon University, Pittsburgh, PA, USA
- Martin Nägele  (1)  
Department of Mathematics,  
ETH Zurich, Zurich, Switzerland
- Minghui Ouyang  (61)  
School of Mathematical Sciences,  
Peking University, China
- Soumit Pal (26)  
Indian Statistical Institute, Kolkata, India

- Katarzyna Paluch  (28)  
University of Wrocław, Wrocław, Poland
- Fahad Panolan  (6)  
University of Leeds, UK
- Marcus Pappik  (56)  
Hasso Plattner Institute, University of Potsdam,  
Germany
- Manaswi Paraashar  (64, 71)  
University of Copenhagen, Denmark
- Michal Parnas (35)  
The Academic College of Tel Aviv-Yaffo,  
Tel Aviv 61083, Israel
- Rafael Pass (66)  
Tel Aviv University, Israel;  
Cornell Tech, New York, NY, USA
- Michał Pawłowski (12)  
MIMUW, University of Warsaw, Poland;  
IDEAS NCBR, Warsaw, Poland
- Will Perkins  (56)  
School of Computer Science, Georgia Institute of  
Technology, Atlanta, GA, USA
- Kalina Petrova  (74)  
ETH Zürich, Department of Computer Science,  
Zürich, Switzerland
- Vladimir Podolskii  (50)  
Tufts University, Medford, MA, USA
- Paweł Prałat (5)  
Department of Mathematics, Toronto  
Metropolitan University, Canada
- Kirk Pruhs  (20)  
Department of Computer Science, University of  
Pittsburgh, Pittsburgh, PA, USA
- Rajmohan Rajaraman  (22)  
Northeastern University,  
Boston MA 02115, USA
- Shravas Rao  (58)  
Portland State University, Portland,  
OR, United States of America
- R. Ravi  (21)  
Carnegie Mellon University, United States
- Connor Riddlesden (39)  
Department of Mathematics and Computer  
Science, Eindhoven University of Technology,  
Netherlands
- Maurice Rolvien (39)  
Department of Computer Science,  
TU Dortmund, Germany
- Matteo Russo (12)  
Sapienza University of Rome, Italy
- Arvin Sahami  (67)  
Department of Computer Science and  
Department of Mathematics, University of  
British Columbia, Vancouver, Canada
- Swagato Sanyal (64)  
Indian Institute of Technology Kharagpur, India
- Nitin Saurabh (64, 71)  
Indian Institute of Technology Hyderabad, India
- Saket Saurabh  (4, 6)  
The Institute of Mathematical Sciences,  
HBNI, Chennai, India;  
University of Bergen, Norway
- Mathias Schacht (59)  
Fachbereich Mathematik, Universität Hamburg,  
Germany
- Kevin Schewior  (14)  
University of Southern Denmark, Odense,  
Denmark
- Leon Schiller (74)  
ETH Zürich, Department of Computer Science,  
Zürich, Switzerland
- Jens Schlöter  (17)  
Faculty of Mathematics and Computer Science,  
University of Bremen, Bremen, Germany
- Sören Schmitt  (10)  
Department of Mathematics,  
University of Siegen, Germany
- Gregory Schwartzman (57)  
JAIST, Nomi, Japan
- Rocco A. Servedio  (52)  
Columbia University, New York, NY, USA
- Jiří Sgall  (8, 10)  
Computer Science Institute of Charles Univ.,  
Faculty of Mathematics and Physics, Prague,  
Czechia
- Hadas Shachnai  (27)  
Computer Science Department,  
Technion, Haifa, Israel
- Vihan Shah (24)  
University of Waterloo, ON, Canada


- Chong Shangguan  (61)  
Research Center for Mathematics and  
Interdisciplinary Sciences, Shandong University,  
China;  
Frontiers Science Center for Nonlinear  
Expectations, Ministry of Education, Qingdao,  
China
- Yuanting Shen  (61)  
Research Center for Mathematics and  
Interdisciplinary Sciences, Shandong University,  
China
- Igor Shinkar  (34)  
Simon Fraser University, Burnaby, Canada
- Harsimran Singh  (34)  
Simon Fraser University, Burnaby, Canada
- Amit Sinhababu (75)  
Chennai Mathematical Institute, Chennai, India
- Youngtak Sohn  (47)  
Department of Mathematics, Massachusetts  
Institute of Technology, USA
- A. R. Sricharan  (40)  
Faculty of Computer Science, Doctoral School  
Computer Science, University of Vienna, Austria
- David Stalfa (22)  
Northeastern University,  
Boston MA 02115, USA
- Teresa Anna Steiner  (40)  
Technical University of Denmark,  
Lyngby, Denmark
- Noah Stephens-Davidowitz (19)  
Cornell University, Ithaca, NY, USA
- Hao Sun  (18)  
University of Alberta, 116 St & 85 Ave,  
Edmonton, AB T6G 2R3, Canada
- Chaitanya Swamy  (29)  
Dept. of Combinatorics and Optimization, Univ.  
Waterloo, Waterloo, ON N2L 3G1, Canada
- Amnon Ta-Shma  (31)  
Department of Computer Science,  
Tel Aviv University, Israel
- Cheng Tan  (22)  
Northeastern University,  
Boston MA 02115, USA
- Thomas Thierauf (75)  
Ulm University, Germany
- Dante Tjowasi (36)  
Cheriton School of Computer Science,  
University of Waterloo, Canada
- Manuel R. Torres  (9)  
University of Illinois, Urbana-Champaign, USA
- Amitabh Trehan  (17)  
Department of Computer Science,  
Durham University, Durham, United Kingdom
- Tomer Tsachor  (15)  
Computer Science Department,  
Technion, Haifa, Israel
- Omer Tubul (60)  
Faculty of Engineering, Bar-Ilan University,  
Ramat Gan, Israel
- Jakub Tětek  (73)  
INSAIT, Sofia, Bulgaria
- Seun William Umboh  (12, 25)  
School of Computing and Information Systems,  
The University of Melbourne, Australia;  
ARC Training Centre in Optimisation  
Technologies, Integrated Methodologies, and  
Applications (OPTIMA), Melbourne, Australia
- Rob van Stee  (10)  
Department of Mathematics,  
University of Siegen, Germany
- N. V. Vinodchandran (26)  
University of Nebraska, Lincoln, USA
- Ben Lee Volk  (44)  
Efi Arazi School of Computer Science,  
Reichman University, Israel
- David Wajc  (13)  
Technion, Haifa, Israel
- Chen Wang (24)  
Rice University, Houston, TX, USA;  
Texas A&M University, College Station,  
TX, USA
- Di Wang  (13)  
Google Research, Mountain View, USA
- Hsin-Po Wang  (65)  
University of California, Berkeley, CA, USA
- Evelyn Warton (42)  
Oregon State University, Corvallis, OR, USA
- Andreas Wiese  (14)  
Technische Universität München, Germany




Anthony Wirth  (25)  
School of Computer Science, The University of  
Sydney, Australia;  
School of Computing and Information Systems,  
The University of Melbourne, Australia


David P. Woodruff  (55)  
Carnegie Mellon University,  
Pittsburgh, PA, USA


Mary Wootters (53)  
Stanford University, CA, USA


Weronika Wrzós-Kaminska  (16)  
EPFL, Lausanne, Switzerland

Ruiyang Wu  (69)  
CFCS, School of CS, Peking University, China


Jie Xue  (6)  
New York University Shanghai, China

Chao Yan  (41)  
Georgetown University, Washington, DC,  
United States of America


Corrine Yap  (56)  
School of Mathematics, Georgia Institute of  
Technology, Atlanta, GA, USA

Yuichi Yoshida  (57)  
Principles of Informatics Research Division,  
National Institute of Informatics, Tokyo, Japan

Huacheng Yu  (33)  
Princeton University, NJ, USA


Ron Zadicario  (31)  
Department of Computer Science,  
Tel Aviv University, Israel

Pavel Zakharov (39)  
Department of Computer Science,  
TU Dortmund, Germany


Meirav Zehavi  (4, 6)  
Ben-Gurion University of the Negev,  
Beer-Sheva, Israel

Xinyuan Zhang (32)  
State Key Laboratory for Novel Software  
Technology, New Cornerstone Science  
Laboratory, Nanjing University, 163 Xianlin  
Avenue, Nanjing, Jiangsu Province, China

Haodong Zhu (39)  
Department of Mathematics and Computer  
Science, Eindhoven University of Technology,  
Netherlands

Weihao Zhu  (9)  
University of Illinois, Urbana-Champaign, USA

Zongrui Zou (32)  
State Key Laboratory for Novel Software  
Technology, New Cornerstone Science  
Laboratory, Nanjing University, 163 Xianlin  
Avenue, Nanjing, Jiangsu Province, China

Stanislav Živný  (7)  
Department of Computer Science,  
University of Oxford, UK



# A $(3/2 + 1/e)$ -Approximation Algorithm for Ordered TSP

Susanne Armbruster ✉ 

Research Institute for Discrete Mathematics and Hausdorff Center for Mathematics,  
University of Bonn, Germany

Matthias Mnich ✉ 

Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Martin Nägele ✉ 

Department of Mathematics, ETH Zurich, Zurich, Switzerland<sup>1</sup>

---

## Abstract

We present a new  $(3/2 + 1/e)$ -approximation algorithm for the Ordered Traveling Salesperson Problem (Ordered TSP). Ordered TSP is a variant of the classic metric Traveling Salesperson Problem (TSP) where a specified subset of vertices needs to appear on the output Hamiltonian cycle in a given order, and the task is to compute a cheapest such cycle. Our approximation guarantee of approximately 1.868 holds with respect to the value of a natural new linear programming (LP) relaxation for Ordered TSP. Our result significantly improves upon the previously best known guarantee of  $5/2$  for this problem and thereby considerably reduces the gap between approximability of Ordered TSP and metric TSP. Our algorithm is based on a decomposition of the LP solution into weighted trees that serve as building blocks in our tour construction.

**2012 ACM Subject Classification** Theory of computation → Discrete optimization; Theory of computation → Routing and network design problems; Theory of computation → Rounding techniques

**Keywords and phrases** Travelling Salesperson Problem, precedence constraints, linear programming, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.1

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2405.06244> [1]

**Funding** *Matthias Mnich:* Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), project MN 59/4-1.

*Martin Nägele:* Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXZ-2047/1 – 390685813, and the Swiss National Science Foundation (grant no. P500PT\_206742).



## 1 Introduction

The classic metric Traveling Salesperson Problem (TSP) is one of the most fundamental and well-studied problems in Combinatorial Optimization and has a large number of applications. A metric TSP instance is given by a complete undirected graph  $G = (V, E)$  with metric edge cost  $c: E \rightarrow \mathbb{R}_{\geq 0}$ . The task is to find a cycle of minimum cost that visits each vertex exactly once, where the cost of a cycle equals the sum of the edge costs over all edges it contains. Metric TSP is highly relevant in many practical applications and thus, a lot of different

---

<sup>1</sup> Most of this work was done while M. Nägele was employed at University of Bonn.

variants are studied (see, e.g., [34]). The problem is NP-hard and APX-hard [32]; concretely, assuming  $P \neq NP$ , it is known that no polynomial-time algorithm can guarantee to find a cycle of cost at most  $1^{23}/122$  times the cost of a cheapest cycle [24]. For a long time, the best-known approximation algorithm for metric TSP was the Christofides-Serdyukov  $3/2$ -approximation algorithm [10, 11, 35]. This was recently improved to a breakthrough  $(3/2 - \varepsilon)$ -approximation algorithm, for some  $\varepsilon > 10^{-36}$ , by Karlin, Klein, and Oveis Gharan [22, 23].

In this work, we focus on a generalization of metric TSP known as *Ordered TSP*, in which some of the vertices must be visited in a given order.

**Ordered TSP (OTSP):** Given a complete undirected graph  $G = (V, E)$  with metric edge cost  $c: E \rightarrow \mathbb{R}_{\geq 0}$  and pairwise distinct vertices  $d_1, \dots, d_k \in V$ , the task is to find a cheapest spanning cycle  $C$  in  $G$  that contains the vertices  $d_1, \dots, d_k$  in this order.

We typically refer to an input of OTSP as an *OTSP instance*  $(G, c, (d_1, \dots, d_k))$ ; solutions are often called *tours*. Our goal in this paper is to further the understanding of the approximability of OTSP, i.e., we aim to design  $\alpha$ -approximation algorithms for OTSP with  $\alpha$  as small as possible.

Clearly, OTSP is at least as hard as metric TSP, and therefore APX-hard. Surprisingly, not much more is known on the approximability of OTSP. Böckenhauer, Hromkovič, Kneis, and Kupke [6] observed that a  $5/2$ -approximate solution can be readily obtained by first traversing  $d_1, \dots, d_k$  in this order and subsequently appending a tour on  $V \setminus \{d_1, \dots, d_k\}$  constructed through the Christofides-Serdyukov algorithm. The black-box use of a metric TSP approximation algorithm allows to reduce this guarantee by the same additive improvement of  $\varepsilon > 10^{-36}$  as in the  $(3/2 - \varepsilon)$ -approximation by Karlin, Klein, and Oveis Gharan. Besides that, Böckenhauer, Mönke, and Steinová [7] gave a  $(5/2 - 2/k)$ -approximation algorithm, where  $k \geq 2$  is the number of ordered vertices in the OTSP input. Note that their result does not directly inherit the improvement achieved for metric TSP, making its approximation ratio asymptotically inferior to the earlier approach of Böckenhauer, Hromkovič, Kneis, and Kupke. Finally, the intuition that OTSP should become easier once  $k$  approaches  $n$  is confirmed by a dynamic programming approach of Deĭneko, Hoffmann, Okamoto, and Woeginger [14] that runs in  $O(2^r r^2 n)$  time and  $O(2^r r n)$  space, i.e., in polynomial time and space if  $r := n - k$ , the number of vertices that are not in the input order, is of magnitude  $O(\log n)$ .

OTSP is in fact a special case of a the following significantly more general TSP variation termed *TSP with Precedence Constraints*.

**TSP with Precedence Constraints (TSP-PC):** Given a complete undirected graph  $G = (V, E)$  with metric edge cost  $c: E \rightarrow \mathbb{R}_{\geq 0}$  and a partial order  $\prec$  on  $V$ , the task is to find a cheapest spanning cycle  $C$  in  $G$  that respects  $\prec$ , i.e.,  $C$  can be traversed such that whenever  $u \prec v$  for two vertices  $u, v \in V$ , then  $u$  appears earlier on  $C$  than  $v$ .

Compared to the total order constraints in OTSP, general partial orders allow for modeling a much wider range of problems. One among many applications of TSP-PC is, e.g., tour planning for mixed pickup and delivery services, where one needs to make sure that a pickup happens before a delivery (but apart from that, pickups and deliveries can be intertwined arbitrarily). There is a considerable body of research on the structure of the

TSP-PC polyhedron, different dynamic programming algorithms, enhanced branch-and-bound methods, and various other exact and heuristic approaches, typically even for the more general version of TSP-PC with asymmetric edge cost (see, e.g., [2, 18, 25, 33] and references therein). Despite that, essentially no positive results on the approximability of TSP-PC are known, which is possibly explained by an influential hardness result of Charikar, Motwani, Raghavan, and Silverstein [9]: By relating the problem to the *Shortest Common Supersequence Problem*, they are able to show that there is no  $(\log n)^\delta$ -approximation for the path version of TSP-PC for any constant  $\delta$ , unless  $\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)})$ , even if the underlying metric space is a line. This motivates our study of the approximability of TSP-PC on general metric spaces with special partial orders, i.e., OTSP.

## 1.1 Our results and techniques

Our main contribution is to significantly improve the state of the art for OTSP by giving an LP-relative approximation guarantee of  $3/2 + 1/e \approx 1.868$ , as stated in the following theorem.

► **Theorem 1.** *There is a polynomial-time  $(3/2 + 1/e)$ -approximation algorithm for OTSP.*

This constitutes a significant improvement over the previous  $(5/2 - \varepsilon)$ -approximation algorithm. We achieve this improvement by introducing a new linear programming (LP) relaxation for OTSP and devising a suitable rounding procedure. The LP relaxation is based on the Held-Karp relaxation that is typically leveraged in the context of TSP, but allows for taking the prescribed order of the vertices  $d_1, \dots, d_k$  into account by using disjoint sets of variables to represent the  $d_i$ - $d_{i+1}$  strolls<sup>2</sup> that a solution is composed of. Our rounding procedure crucially relies on a result on decomposing (fractional)  $s$ - $t$  strolls into a convex combination of trees. This decomposition resembles an existential result by Bang-Jensen, Frank, and Jackson [3, Theorem 2.6] on packing branchings in a directed multigraph. Variations thereof have recently been used for advances on another variant of TSP, namely *Prize-Collecting TSP* [4, 5], and motivate the application here. (See Lemma 5 for the precise statement of the decomposition result.) The trees obtained from stroll decompositions enable the construction of a subgraph that spans a reasonably large part of  $V$  at cost no more than the LP solution cost, and contains a walk with visits at  $d_1, \dots, d_k$  in this order. Our tour construction is completed by connecting the remaining isolated vertices in a cheapest possible way, and applying a parity correction step as typical for TSP-like problems.

Our approach crucially relies on being able to split a solution into  $d_i$ - $d_{i+1}$  strolls upfront, hence it is not directly suitable for handling arbitrary precedence constraints other than total orders. While one can always try to guess a suitable total order that is compatible with the given partial order, and then apply Theorem 1, this is generally not efficient. We can, though, obtain approximation algorithms for some special cases of precedence constraints, as for example in the following result that is a direct generalization of Theorem 1.

► **Theorem 2.** *Consider a TSP-PC instance  $(G, c, \prec)$  on a complete graph  $G = (V, E)$  with a partial order  $\prec$  that can be equivalently given as independent total orders on disjoint subsets  $D_1, \dots, D_\ell \subseteq V$ . There is a polynomial-time  $(\ell + 1/2 + 1/e^\ell)$ -approximation algorithm for this class of TSP-PC problems.*

<sup>2</sup> We use the term *s-t stroll* instead of *s-t path* for a path from  $s$  to  $t$  in the underlying graph to emphasize that we do not require all vertices to be covered. Also, for convenience of notation, we use  $d_{k+1} := d_1$  throughout the paper.

The total orders on the sets  $D_i$  are also called *chains*. We remark that losing a factor of  $\ell$  in Theorem 2 is intrinsic to our approach: We never merge the given chains, but traverse them one after another. Still, the result of Theorem 2 is superior to a black-box algorithm that independently applies the algorithm from Theorem 1 to the  $\ell$  chains and concatenates the resulting tours (while shortcutting to avoid repeated visits).

## 1.2 Related Work

Variations of OTSP and TSP-PC are also studied in the context of scheduling with precedence constraints. In a classic setup, denoted by  $Pm|prec|C_{\max}$  in the scheduling literature, one needs to find a schedule for a set  $\mathcal{J}$  of  $n$  jobs on  $m$  identical machines subject to precedence constraints between the jobs. Formally, each job  $j \in \mathcal{J}$  is characterized by a processing time  $p_j \in \mathbb{Z}_{\geq 0}$ , and a schedule  $\sigma: \mathcal{J} \rightarrow \mathbb{Z}_{\geq 0} \times \{1, \dots, m\}$  assigns each job  $j \in \mathcal{J}$  to a pair  $(\sigma_1(j), \sigma_2(j))$  consisting of an integer start time  $\sigma_1(j)$  and a machine  $\sigma_2(j)$  such that no other job scheduled on that machine has their start time in the time interval  $[\sigma_1(j), \sigma_1(j) + p_j]$ , and for any two jobs  $j, j' \in \mathcal{J}$  related as  $j \prec j'$  it holds that  $\sigma_1(j) < \sigma_2(j')$ . The makespan objective  $C_{\max}$  of a schedule  $\sigma$  is the maximum completion time  $C_j = \sigma_1(j) + p_j$  over all jobs  $j \in \mathcal{J}$ . Generally, precedence constraints of this type are studied extensively in a wide range of scheduling problems, including different settings and objectives (see, e.g., [13, 19, 28, 36]). The three-machine problem  $P3|prec, p_j \equiv 1|C_{\max}$  is one of the few famous open problems by Garey and Johnson [17] whose computational complexity has not yet been resolved.

The complexity of many scheduling problems with precedence constraints that are chains has been well-investigated. An influential paper of Lenstra and Rinnooy Kan [27] shows strong NP-hardness for minimizing the number of chain-constrained unit-size jobs that miss their deadline on a single machine. Several other works with chain constraints have appeared [15, 21, 26, 37].

Towards analogues of TSP-PC, we may consider the problem  $Pm|prec|C_{\max}$  on a single machine, but add sequence-dependent setup times  $s_{ij} \in \mathbb{Z}_{\geq 0}$  between any two jobs  $i$  and  $j$ , which add to the makespan of the schedule. This problem, which is denoted as  $1|prec, s_{ij}|C_{\max}$ , was discussed by Liaee and Emmons [29, Section 3.1.2]. In case of TSP-PC, the setup times are metric (i.e.,  $s_{ij} \leq s_{ik} + s_{kj}$  for any triple  $(i, j, k)$  of distinct jobs), and all jobs have equal processing time  $p_j \equiv 0$ . To be precise, the objective function for TSP-PC takes into account the cost for returning to the origin city whereas no such cost occurs in the objective function for  $1|prec, s_{ij}|C_{\max}$ , hence the latter in fact models a path version of TSP-PC.

We remark that shortly after the first dissemination of this work [1], Böhm, Friggstad, Mömke, and Spoerhase [8] provided an independent paper on TSP variants, including a result matching the guarantee in Theorem 1. Albeit different at first sight, their approach is very similar at its core to the one presented here: They use a directed analogue of the LP relaxation that we also build our work on here, and thereby facilitate a direct application of the original result by Bang-Jensen, Frank, and Jackson on packing branchings in a directed multigraph.

## 1.3 Organization of the paper

In Section 2, we introduce our new linear programming formulation for OTSP (Section 2.1) and analyze a randomized algorithm giving the guarantee of Theorem 1 in expectation (Section 2.2). We show how this algorithm can be derandomized in Section 2.3. Finally, Section 3 extends our framework to yield Theorem 2, and Section 4 shows how our main technical lemma is implied by a closely related known result.

## 2 Our algorithm

### 2.1 The LP relaxation and polyhedral basics

The most commonly used LP relaxation in approximation algorithms for classic TSP is the so-called *Held-Karp relaxation*. It was first introduced by Dantzig, Fulkerson, and Johnson [12] and is given by

$$P_{\text{HK}}(G) := \left\{ x \in \mathbb{R}_{\geq 0}^E : \begin{array}{l} x(\delta(v)) = 2 \quad \forall v \in V \\ x(\delta(S)) \geq 2 \quad \forall S \subsetneq V, S \neq \emptyset \end{array} \right\},$$

where  $G = (V, E)$  is the underlying complete graph.<sup>3</sup> While TSP simply asks for a spanning cycle, OTSP requires that the vertices  $d_1, \dots, d_k$  appear on the cycle in this order. Thus, a solution is naturally composed of  $k$  strolls, namely a  $d_i$ - $d_{i+1}$  stroll for every  $i \in \{1, \dots, k\}$ . For a polyhedral description of  $s$ - $t$  strolls in a complete graph  $G = (V, E)$ , we modify the Held-Karp relaxation for  $s$ - $t$  path TSP<sup>4</sup> to allow partial coverage of vertices. Concretely, the variables  $y \in \mathbb{R}_{\geq 0}^V$  in the following formulation indicate the extent at which vertices are covered.<sup>5</sup>

$$P_{s-t \text{ stroll}}(G) := \left\{ (x, y) \in \mathbb{R}_{\geq 0}^E \times \mathbb{R}_{\geq 0}^V : \begin{array}{l} x(\delta(v)) = 2y_v \quad \forall v \in V \\ x(\delta(S)) \geq 1 \quad \forall S \subseteq V \setminus \{t\}, s \in S \\ x(\delta(S)) \geq 2y_v \quad \forall S \subseteq V \setminus \{s, t\}, v \in S \\ y_s = y_t = 1/2 \end{array} \right\}. \quad (1)$$

Note that setting  $y_s = y_t = 1/2$  corresponds to  $s$  and  $t$  having degree 1 in an  $s$ - $t$  stroll, while all interior vertices of an integral stroll have degree 2, which corresponds to a  $y$ -value of 1. Using the above polyhedral relaxation (1) for all  $d_i$ - $d_{i+1}$  strolls, it remains to link the strolls by requiring full joint coverage of every  $v \in V$ . This results in the following LP relaxation for OTSP:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \sum_{i=1}^k x_e^i \\ & \sum_{i=1}^k y_v^i = 1 \quad \forall v \in V \\ & (x^i, y^i) \in P_{d_i-d_{i+1} \text{ stroll}}(G) \quad \forall i \in \{1, \dots, k\}. \end{aligned} \quad (\text{OTSP LP relaxation})$$

Here and throughout the paper, note that we use superscripts to distinguish different strolls. It is clear that any OTSP solution can be turned into a feasible solution to the above LP of the same objective value, hence the above LP is indeed a relaxation of OTSP. We first observe that this OTSP LP relaxation strengthens the Held-Karp relaxation in the following sense.

► **Observation 3.** *Let  $(x^i, y^i)_{i \in \{1, \dots, k\}}$  be feasible for the OTSP LP relaxation. Then  $x := \sum_{i=1}^k x^i \in P_{\text{HK}}(G)$ .*

<sup>3</sup> For  $S \subseteq V$  we denote by  $\delta(S)$  the set of edges with exactly one endpoint in  $S$ . For  $v \in V$ , we abbreviate  $\delta(v) := \delta(\{v\})$ .

<sup>4</sup> Given a complete graph  $G = (V, E)$  with metric edge costs and vertices  $s, t \in V$ ,  $s$ - $t$  path TSP is the variant of TSP that seeks a path of smallest total cost from  $s$  to  $t$  while visiting every vertex exactly once.

<sup>5</sup> The constraints of  $P_{s-t \text{ stroll}}$  imply that for  $v \in V \setminus \{s, t\}$ , we have  $2y_v \leq x(\delta(V \setminus \{s, t\})) \leq x(\delta(s)) + x(\delta(t)) \leq 2$ , and thus  $y_v \leq 1$ , legitimating the proposed interpretation.

**Proof.** To see that  $x$  satisfies the degree constraints in  $P_{\text{HK}}(G)$ , note that for all  $v \in V$ , we have

$$x(\delta(v)) = \sum_{i=1}^k x^i(\delta(v)) = 2 \cdot \sum_{i=1}^k y_i = 2 .$$

To verify the cut constraints, let  $S \subsetneq V$  be a non-empty set of vertices. If both  $S \cap \{d_1, \dots, d_k\} \neq \emptyset$  and  $(V \setminus S) \cap \{d_1, \dots, d_k\} \neq \emptyset$ , then there exist two distinct indices  $i_1, i_2 \in \{1, \dots, k\}$  such that  $d_{i_1} \in S$  but  $d_{i_1+1} \notin S$ , and  $d_{i_2} \notin S$  but  $d_{i_2+1} \in S$ . This implies that  $x^{i_1}(\delta(S)) \geq 1$  and  $x^{i_2}(\delta(V \setminus S)) \geq 1$ , so we get

$$x(\delta(S)) = \sum_{i=1}^k x^i(\delta(S)) \geq x^{i_1}(\delta(S)) + x^{i_2}(\delta(S)) = x^{i_1}(\delta(S)) + x^{i_2}(\delta(V \setminus S)) \geq 2 .$$

Otherwise, assume without loss of generality that  $S \cap \{d_1, \dots, d_k\}$  is empty (if not,  $V \setminus S$  has this property) and fix a vertex  $v \in S$ . We then know that  $x^i(\delta(S)) \geq 2y_v$  for all  $i \in \{1, \dots, k\}$ , hence

$$x(\delta(S)) = \sum_{i=1}^k x^i(\delta(S)) \geq 2 \cdot \sum_{i=1}^k y_v = 2 . \quad \blacktriangleleft$$

The point  $x \in P_{\text{HK}}(G)$  constructed in Observation 3 has the property that its cost  $c^\top x$  equals the objective value  $c_{\text{LP}}$  of the feasible point of the OTSP LP relaxation that we started with. Thus, following the arguments of Wolsey's polyhedral analysis [38] of the Christofides-Serdyukov algorithm, we immediately obtain the following.

► **Corollary 4.** *Let  $c_{\text{LP}}$  denote the optimal objective value of the OTSP LP relaxation. Then in the underlying graph  $G$  with edge costs  $c$ , the following holds true.*

- (i) *A shortest spanning tree  $T$  satisfies  $c(T) \leq c_{\text{LP}}$ .*
- (ii) *For any even cardinality set  $Q \subseteq V$ , a shortest  $Q$ -join  $J$  satisfies  $c(J) \leq \frac{1}{2} \cdot c_{\text{LP}}$ .*

**Proof.** Let  $(x^i, y^i)_{i \in \{1, \dots, k\}}$  be an optimal solution of the OTSP LP relaxation. By Observation 3,  $x := \sum_{i=1}^k x^i \in P_{\text{HK}}(G)$ . It is well-known due to Held and Karp [20] that then,  $\frac{|V|-1}{|V|} \cdot x$  is feasible for the spanning tree polytope, and due to Wolsey [38] that  $\frac{1}{2}x$  is feasible for the dominant of the  $Q$ -join polytope, hence  $c(T) \leq \frac{|V|-1}{|V|} \cdot c^\top x < c^\top x$  and  $c(J) \leq \frac{1}{2}c^\top x$ . Using that  $c^\top x = c_{\text{LP}}$ , the result follows. ◀

## 2.2 Rounding an LP solution

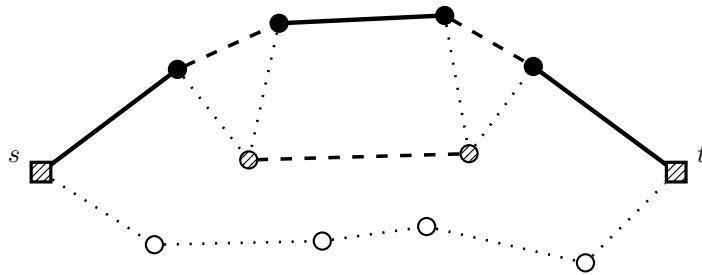
At its core, our algorithm for rounding a typically fractional solution  $(x^i, y^i)_{i \in \{1, \dots, k\}}$  of the OTSP LP relaxation is based on leveraging a decomposition result for each of the points  $(x^i, y^i) \in P_{d_i-d_{i+1} \text{ stroll}}$ . By scaling up  $(x^i, y^i)$  by a large enough factor  $M$  such that  $Mx^i$  is integral, this decomposition can be viewed as a result on packing trees into the multigraph that has  $Mx_e^i$  copies of every edge  $e \in E$  and such that every vertex  $v$  appears in  $My^i$  many of the trees. While most results of this type deal with packing spanning trees (or, in the directed case, arborescences), i.e., consider uniform packings, Bang-Jensen, Frank, and Jackson [3] gave one of few results in a non-uniform setting as we are facing here. Their splitting-off based construction was revised by Blauth and Nägele [5] to obtain more fine-grained control over the output components of the decomposition when starting from a solution of a Held-Karp-type relaxation that allows partial coverage of vertices (similar to what we allow in  $P_{s-t \text{ stroll}}$ ). We observe that these findings can be immediately carried over to solutions of  $P_{s-t \text{ stroll}}$ , giving Lemma 5 below. We defer a formal proof to Section 4.



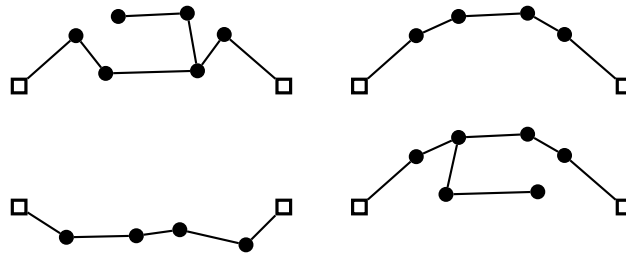
► **Lemma 5.** *Let  $G = (V, E)$  be an undirected graph,  $s, t \in V$ , and let  $(x, y) \in P_{s-t \text{ stroll}}(G)$ . We can in polynomial time compute a family  $\mathcal{T}$  of subtrees of  $G$  that each contain the vertices  $s$  and  $t$ , and weights  $\mu \in [0, 1]^{\mathcal{T}}$  with  $\sum_{T \in \mathcal{T}} \mu_T = 1$  such that<sup>6</sup>*

$$\sum_{T \in \mathcal{T}} \mu_T \chi^{E[T]} = x \quad \text{and} \quad \sum_{T \in \mathcal{T}: v \in V[T]} \mu_T = y_v \quad \forall v \in V \setminus \{s, t\} .$$

In other words, Lemma 5 allows to decompose a fractional  $s$ - $t$  stroll into a convex combination of trees in a family  $\mathcal{T}$  that all connect  $s$  and  $t$ , and such that for every other vertex  $v \in V \setminus \{s, t\}$ , the weighted number of trees that contain  $v$  equals the coverage  $y_v$  of  $v$  in the stroll. An example of a feasible solution  $(x, y)$  and a decomposition satisfying the properties of Lemma 5 is given in Figure 1.



(a) Solution  $(x, y)$  with  $x_e = 1/4$  for dotted edges,  $x_e = 1/2$  for dashed edges, and  $x_e = 3/4$  for solid edges. Likewise,  $y_v = 1/4$  for blank vertices,  $y_v = 1/2$  for dashed vertices, and  $y_v = 3/4$  for full vertices.



(b) A decomposition of the solution  $(x, y)$  given in (a) into four trees with uniform weight  $\mu \equiv 1/4$  satisfying the properties of Lemma 5.

■ **Figure 1** A solution  $(x, y) \in P_{s-t \text{ stroll}}$  along with a decomposition into trees, exemplifying Lemma 5.

After applying Lemma 5 to all strolls  $(x^i, y^i) \in P_{d_i-d_{i+1} \text{ stroll}}$  obtained from an optimal solution of the OTSP LP relaxation, we choose one tree from each of the decompositions and consider the (multi-)union of all edges obtained this way. This results in a graph that already contains a closed walk with visits at  $d_1, \dots, d_k$  in this order, giving the basis for our construction of an OTSP solution. Also, we can easily bound the expected cost of the edge set obtained in this way by randomly choosing the trees with marginals given by the weights  $\mu$  from Lemma 5.

<sup>6</sup> For a graph  $H$  we denote by  $V[H]$  the set of vertices and by  $E[H]$  the set of edges of  $H$ . Furthermore, for an edge set  $F \subseteq E$ , we denote by  $\chi^F \in \{0, 1\}^E$  the characteristic vector of  $F$ , i.e., the  $\{0, 1\}$ -vector with, for all  $e \in E$ ,  $\chi_e^F = 1$  if and only if  $e \in F$ .

To obtain an actual OTSP solution, the missing steps are to

- (i) connect vertices that are not covered by any of the trees  $T_i$ ,
- (ii) perform parity correction to guarantee that there exists an Eulerian tour, and
- (iii) shortcut appropriately to obtain an actual OTSP solution.

Altogether, this leads to the randomized Algorithm 1 as laid out below; also see Figure 2 for an example illustration of the different edge sets that are constructed in Algorithm 1.

■ **Algorithm 1** A randomized approximation algorithm for OTSP.

---

**Input:** OTSP instance  $(G, c, \{d_1, \dots, d_k\})$  on graph  $G = (V, E)$ .

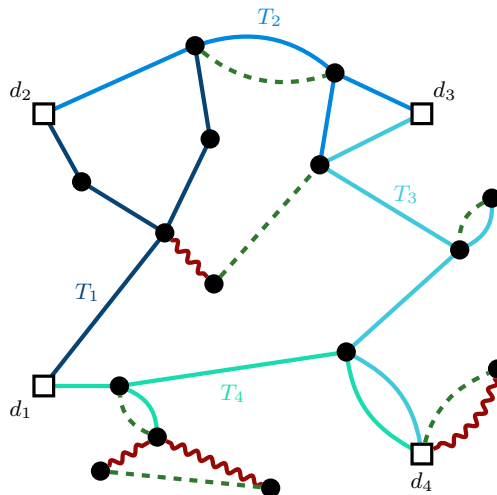
- 1 Compute an optimal solution  $(x^i, y^i)_{i \in \{1, \dots, k\}}$  to the OTSP LP relaxation.
- 2 **foreach**  $i \in \{1, \dots, k\}$  **do**
- 3     Apply Lemma 5 to decompose  $(x^i, y^i)$  into trees  $\mathcal{T}_i$  with weights  $\mu^i$ .
- 4     Sample one tree  $T_i$  from  $\mathcal{T}_i$  with marginals given by  $\mu^i$ .
- 5 Compute a minimum-cost edge set  $F \subseteq E$  such that the multigraph

$$H := \left( V, F \cup \bigcup_{i \in \{1, \dots, k\}} E[T_i] \right)$$

is connected.

- 6 Let  $Q = \text{odd}(H)$  and compute a minimum cost  $Q$ -join  $J$  in  $G$ .
  - 7 **return** Spanning cycle  $C$  in  $G$  obtained from  $H \dot{\cup} J$  through Lemma 10.
- 

We first show that Algorithm 1 gives the guarantees claimed by Theorem 1 in expectation and – even stronger – with respect to the value  $c_{\text{LP}}$  of the OTSP LP relaxation, as stated in the subsequent theorem. In Section 2.3, we show that Algorithm 1 admits an immediate derandomization using the method of conditional expectation, thereby completing the proof of Theorem 1.



■ **Figure 2** Exemplifying the construction of Eulerian graph  $H \dot{\cup} J$  from Algorithm 1: Trees  $T_1, T_2, T_3, T_4$  drawn as solid blueish edges, the edge set  $F$  connecting all vertices to the trees drawn as curly red edges, and the  $\text{odd}(H)$ -join  $J$  drawn as dashed green edges.

► **Theorem 6.** *Let  $c_{LP}$  be the cost of an optimal solution of the OTSP LP relaxation. Algorithm 1 returns in polynomial time an OTSP solution  $C$  satisfying*

$$\mathbb{E}[c(E[C])] \leq \left(\frac{3}{2} + \frac{1}{e}\right) \cdot c_{LP} .$$

To prove Theorem 6, we first study the random graph  $H_0 := (V, \dot{\bigcup}_{i \in \{1, \dots, k\}} E[T_i])$  obtained from taking the union of trees  $T_i \in \mathcal{T}_i$  for all  $i \in \{1, \dots, k\}$  as sampled in Algorithm 1. In order for the following statements to also be applicable in a proof of Theorem 2, we refer to the tree distributions of the type generated in Algorithm 1 as *connecting tree distributions*.

► **Definition 7** (Connecting tree distribution). *Let  $G = (V, E)$  be a graph and let  $d_1, \dots, d_k \in V$ . A connecting tree distribution  $(\mathcal{T}_i, \mu^i)_{i \in \{1, \dots, k\}}$  consists of a family  $\mathcal{T}_i$  of subtrees of  $G$  and marginals  $\mu^i: \mathcal{T}_i \rightarrow (0, 1]$  for every  $i \in \{1, \dots, k\}$  with the following properties.*

- (i)  $\sum_{T \in \mathcal{T}_i} \mu_T^i = 1$  for all  $i \in \{1, \dots, k\}$ .
- (ii)  $V[T] \cap \{d_1, \dots, d_k\} = \{d_i, d_{i+1}\}$  for all  $T \in \mathcal{T}_i$  and  $i \in \{1, \dots, k\}$ .
- (iii)  $\sum_{i=1}^k \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i = 1$  for all  $v \in V \setminus \{d_1, \dots, d_k\}$ .

The distributions  $(\mathcal{T}_i, \mu^i)$  obtained in Algorithm 1 by applying Lemma 5 indeed satisfy the constraints of the above definition; in particular, Item iii is fulfilled because

$$\sum_{i=1}^k \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i = \sum_{i=1}^k y_v^i = 1 \quad \forall v \in V \setminus \{d_1, \dots, d_k\} ,$$

where the first equality follows from Lemma 5, and the second one is implied by constraints of  $P_{d_i-d_{i+1}}^{\text{stroll}}$ .

► **Lemma 8.** *Let  $G = (V, E)$  be a graph,  $d_1, \dots, d_k \in V$ , and let  $(\mathcal{T}_i, \mu^i)$  be a connecting tree distribution.*

- (i) *For any choice of trees  $T_i \in \mathcal{T}_i$  for  $i \in \{1, \dots, k\}$ , the multigraph  $H_0 := (V, \dot{\bigcup}_{i \in \{1, \dots, k\}} E[T_i])$  consists of one large connected component and potentially several isolated vertices. The large connected component contains a walk with visits at  $d_1, \dots, d_k$  in this order that can be constructed efficiently from the trees  $T_i$ .*
- (ii) *If, in the above construction, the trees  $T_i$  are sampled with marginals  $\mu^i$ , we have that for all  $v \in V \setminus \{d_1, \dots, d_k\}$ ,*

$$\mathbb{P}[v \text{ is isolated in } H_0] \leq \frac{1}{e} .$$

**Proof.** For Item i observe that each tree  $T_i$  is connected within itself by definition and, as it contains  $d_i$  and  $d_{i+1}$ , the union of all trees form one large connected component, while all other components must be isolated vertices. Also, because each tree  $T_i$  contains a  $d_i-d_{i+1}$  path, we may concatenate these paths to obtain the desired walk with visits at  $d_1, \dots, d_k$  in this order.

To prove Item ii, we calculate the probability that a vertex  $v \in V \setminus \{d_1, \dots, d_k\}$  is isolated in  $H_0$ . First, note that for any such vertex  $v$  and any  $i \in \{1, \dots, k\}$ , we have

$$\mathbb{P}[v \notin V[T_i]] = 1 - \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i .$$

**1:10 A  $(3/2 + 1/e)$ -Approximation Algorithm for Ordered TSP**

Thus, the probability that a vertex  $v \in V \setminus \{d_1, \dots, d_k\}$  is not contained in any tree  $T_i$  for  $i \in \{1, \dots, k\}$ , and hence is isolated in  $H_0$ , can be bounded as follows:

$$\begin{aligned} \mathbb{P}\left[v \notin \bigcup_{i=1}^k V[T_i]\right] &= \prod_{i=1}^k \mathbb{P}[v \notin V[T_i]] = \prod_{i=1}^k \left(1 - \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i\right) \\ &\leq \exp\left(-\sum_{i=1}^k \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i\right) = \frac{1}{e}, \end{aligned}$$

where we used that  $1 - t \leq \exp(-t)$  for all  $t \in \mathbb{R}$ , and  $\sum_{i=1}^k \sum_{T \in \mathcal{T}_i: v \in V[T]} \mu_T^i = 1$  because  $(\mathcal{T}_i, \mu^i)$  is a connecting tree distribution.  $\blacktriangleleft$

Next, we bound the cost of the minimum-cost connector  $F$  computed in Line 1 of Algorithm 1.

► **Lemma 9.** *Let  $G = (V, E)$  be a graph with metric edge costs  $c$ , let  $d_1, \dots, d_k \in V$ , and let  $T$  be a minimum-cost spanning tree of  $(G, c)$ .*

- (i) *For all  $v \in V \setminus \{d_1\}$ , let  $e_v$  denote the unique edge outgoing of  $v$  when orienting  $T$  towards  $d_1$ . For every graph  $H_0$  on the vertex set  $V$  with components that are – up to possibly the component containing  $d_1$  – singleton vertices, the minimum-cost edge set  $F$  that connects  $H_0$  satisfies*

$$c(F) \leq \sum_{v \text{ isolated in } H_0} c(e_v).$$

- (ii) *Let  $(\mathcal{T}_i, \mu^i)$  for  $i \in \{1, \dots, k\}$  be a connecting tree distribution. If the trees  $T_i \in \mathcal{T}_i$  are sampled with marginals  $\mu^i$  and  $H_0 := (V, \bigcup_{i \in \{1, \dots, k\}} E[T_i])$ , we obtain*

$$\mathbb{E}[c(F)] \leq \frac{1}{e} c(T).$$

**Proof.** In order to prove Item i, we construct a feasible connecting edge set  $F'$  as the set of all edges  $e_v$  for which  $v$  is an isolated vertex. Then  $H_0 \cup F'$  is indeed connected, because each isolated vertex of  $H_0$  is connected to its predecessor in  $T$  by an edge of  $F'$ , hence inductively, the component of  $H_0$  containing  $d_1$  can be reached along edges of  $F'$ . As the minimum-cost connector  $F$  has cost at most  $c(F')$ , we have

$$c(F) \leq c(F') \leq \sum_{v \text{ isolated in } H_0} c(e_v).$$

To prove Item ii, we note that in this case,  $H_0$  consists of one large connected component and some isolated vertices by Item i of Lemma 8. Using Item ii of Lemma 8 on top of the above, we get

$$\mathbb{E}[c(F)] \leq \mathbb{E}[c(F')] = \sum_{v \in V \setminus \{d_1\}} \mathbb{P}[v \text{ isolated in } H_0] \cdot c(e_v) \leq \frac{1}{e} \sum_{v \in V \setminus \{d_1\}} c(e_v) = \frac{1}{e} c(E[T]). \blacktriangleleft$$

The cost of the odd( $H$ )-join  $J$  constructed in Line 1 of Algorithm 1 can be bounded by  $\frac{1}{2}c^\top x$  by Item ii of Corollary 4. Hence, to complete the analysis of Algorithm 1, it is left to show that from the Eulerian graph  $H \cup J$  constructed in Line 1 of Algorithm 1, we can obtain an OTSP solution of no larger cost. We remark that such a step has also been used by Böckenhauer, Mömke, and Steinová [7]; we repeat it here explicitly and give a slightly

different proof for completeness. In the proof, we repeatedly use the operation of *shortcutting* a vertex  $v$  on a walk, which is the following: If the predecessor and successor of  $v$  on the walk are  $u$  and  $w$ , respectively, we delete the edges  $\{u, v\}$  and  $\{v, w\}$  from the walk and add the direct edge  $\{u, w\}$  instead. It is clear that this operation results in a walk again; furthermore, by the triangle inequality, the costs of the walk do not increase under such operations.

► **Lemma 10.** *Let  $G = (V, E)$  be a complete graph with metric edge costs  $c$ , and let  $d_1, \dots, d_k \in V$  be distinct. Given an undirected connected Eulerian multigraph  $M = (V, E_M)$  together with a closed walk in  $M$  with visits at  $d_1, \dots, d_k$  in this order, we can in polynomial time determine a spanning cycle  $C$  in  $G$  with visits at  $d_1, \dots, d_k$  in this order of cost at most  $c(E_M)$ .*

**Proof.** Let  $C$  be the given closed walk on which  $d_1, \dots, d_k$  appear in this order, delete  $C$  from  $M$  and partition the remaining Eulerian graph into a set  $\mathcal{W}$  of closed walks. Shortcut  $C$  to a cycle while maintaining visits at  $d_1, \dots, d_k$  in this order. This can, for example, be done by traversing  $C$  starting at  $d_1$ , and shortcutting

- (i) vertices that have already been visited, and
- (ii) vertices  $d_i$  that are not yet to be visited due to the order constraint.

Afterwards, as long as  $\mathcal{W}$  is non-empty, pick a closed walk  $W$  from  $\mathcal{W}$  that intersects  $C$ , and let  $v$  be a vertex in the intersection. Traversing  $W$  starting from  $v$ , shortcut  $W$  to a cycle by skipping, except for  $v$  itself, all vertices that are already contained in  $C$ . Then, merge  $W$  into  $C$  by first traversing  $C$  up to (and including)  $v$ , then completely traversing  $W$  until (but not including)  $v$  before continuing on  $C$ , thereby including only one visit at  $v$  in the updated  $C$ . It is immediate that  $C$  is still a cycle after any such operation, and the vertices  $d_1, \dots, d_k$  still appear on  $C$  once and in this order. By connectivity of  $M$ , this procedure only terminates once  $\mathcal{W}$  is empty, and in that case,  $C$  is a spanning cycle of  $G$ . Also, all steps can be implemented to run in polynomial time. Clearly, the final length of  $C$  with respect to  $c$  is at most  $c(E_M)$  because  $c$  is metric. ◀

From the above ingredients, we can readily prove Theorem 6.

**Proof of Theorem 6.** The solution returned by Algorithm 1 is a spanning cycle  $C$  in  $G$  obtained from  $H \dot{\cup} J$  through Lemma 10, hence it is feasible and of cost at most  $c(E[H \dot{\cup} J])$ . Note that the required closed walk in  $H \dot{\cup} J$  with visits at  $d_1, \dots, d_k$  in this order is guaranteed and can be constructed efficiently from the trees  $T_i$  by Item i of Lemma 8. Furthermore, by Item ii of Lemma 9 and Corollary 4, we know that  $\mathbb{E}[c(F)] \leq 1/e \cdot c(T) \leq 1/e \cdot c_{\text{LP}}$ , where  $T$  is a minimum-cost spanning tree. In addition, Corollary 4 also implies that  $c(E[J]) \leq \frac{1}{2} \cdot c_{\text{LP}}$ . Last but not least, we can express the expected cost of each  $T_i$  as

$$\mathbb{E}[c(E[T_i])] = \sum_{T \in \mathcal{T}_i} \mu_T^i c(E[T]) = \sum_{T \in \mathcal{T}_i} \mu_T^i c^\top \chi^{E[T]} = c^\top x^i \quad \forall i \in \{1, \dots, k\} .$$

Thus, by summing over all constructed trees, we obtain  $\sum_{i=1}^k \mathbb{E}[c(E[T_i])] = \sum_{i=1}^k c^\top x^i = c_{\text{LP}}$ . Together, this yields the proclaimed bound

$$\mathbb{E}[c(C)] \leq \mathbb{E}[c(E[H \dot{\cup} J])] \leq \left( \frac{3}{2} + \frac{1}{e} \right) \cdot c_{\text{LP}} .$$

It remains to note that Algorithm 1 can be implemented to run in polynomial time. To start with, an optimal solution of the OTSP LP relaxation can be found in polynomial time because  $P_{s-t \text{ stroll}}$  admits a polynomial-time separation oracle through polynomially many calls to a minimum-cut algorithm. Next, the decomposition in Line 1 is obtained in

polynomial time, finding an optimal edge set  $F$  in Line 1 can be implemented by Prim's algorithm, and the odd( $H$ )-join is well-known to be computable in polynomial time. Finally, also the computation of the cycle  $C$  in Line 1 is polynomial due to Lemma 10, concluding the proof.  $\blacktriangleleft$

### 2.3 Derandomizing Algorithm 1

To complete a proof of our main result, Theorem 1, we now show how to derandomize Algorithm 1 using the method of conditional expectations, which results in the following proof.

**Proof of Theorem 1.** By the construction of the solution  $C$  in Algorithm 1, using Item i of Lemma 9 to bound the cost of  $F$ , and Item ii of Corollary 4 to bound the cost of  $J$ , we know that

$$\begin{aligned} c(C) &\leq \sum_{i=1}^k c(E[T_i]) + c(F) + c(E[J]) \\ &\leq \underbrace{\sum_{i=1}^k c(E[T_i]) + \sum_{v \notin \bigcup_{i=1}^k V[T_i]} c(e_v)}_{=:g(T_1, \dots, T_k)} + \frac{1}{2} \cdot c_{\text{LP}} \ , \end{aligned} \quad (2)$$

where we recall that  $e_v$ , for  $v \in V \setminus \{d_1\}$ , is the unique outgoing edge at  $v$  when orienting a minimum-cost spanning tree of  $G$  towards  $d_1$ . For Theorem 6, we showed that  $\mathbb{E}[g(T_1, \dots, T_k)] \leq (3/2 + 1/e) \cdot c_{\text{LP}}$ . Following the method of conditional expectations, in order to derandomize the choices of the trees  $T_i$  in Line 1 of Algorithm 1 while maintaining the upper bound on the solution cost, we sequentially choose trees  $S_i$  for  $i \in \{1, \dots, k\}$  such that

$$S_i = \arg \min_{S \in \mathcal{T}_i} \mathbb{E}[g(T_1, \dots, T_k) \mid T_1 = S_1, \dots, T_{i-1} = S_{i-1}, T_i = S] \ . \quad (3)$$

Note that feasibility of the cycle  $C$  and the bound of (2) on its cost are unaffected by fixing  $T_i = S_i$ . By definition of conditional expectation, we know that

$$\begin{aligned} \mathbb{E}[g(T_1, \dots, T_k) \mid T_1 = S_1, \dots, T_{i-1} = S_{i-1}] \\ = \sum_{S \in \mathcal{T}_i} \mu_S^i \cdot \mathbb{E}[g(T_1, \dots, T_k) \mid T_1 = S_1, \dots, T_{i-1} = S_{i-1}, T_i = S] \ , \end{aligned}$$

hence the sequence of conditional expectations

$$(\mathbb{E}[g(T_1, \dots, T_k) \mid T_1 = S_1, \dots, T_i = S_i])_{i \in \{1, \dots, k\}}$$

is non-increasing by the choice in (3), because  $\sum_{S \in \mathcal{T}_i} \mu_S = 1$ . Thus, it remains to observe that the conditional expectations in (3) can be computed. To this end, observe that

$$\begin{aligned} \mathbb{E}[g(T_1, \dots, T_k) \mid T_1 = S_1, \dots, T_\ell = S_\ell] \\ = \sum_{i=1}^{\ell} c(E[S_i]) + \sum_{i=\ell+1}^k \mathbb{E}[c(E[T_i])] + \sum_{v \notin \bigcup_{i=1}^{\ell} V[S_i]} \mathbb{P}\left[v \notin \bigcup_{i=\ell+1}^k V[T_i]\right] c(e_v) + \frac{1}{2} \cdot c_{\text{LP}} \ , \end{aligned}$$

and we can readily compute

$$\mathbb{E}[c(E[T_i])] = \sum_{T \in \mathcal{T}_i} \mu_T^i c(E[T]) \quad \text{and} \quad \mathbb{P}\left[v \notin \bigcup_{i=\ell+1}^k V[T_i]\right] = \prod_{i=\ell+1}^k (1 - y_v^i) \ . \quad \blacktriangleleft$$

### 3 Extending to several independent total orders: Proving Theorem 2

In this section, we show how our approach can be extended to TSP-PC with a specific structure of precedence constraints that corresponds to having total orders on disjoint subsets  $D_1, \dots, D_\ell \subseteq V$  of the input graph  $G = (V, E)$ .

As mentioned in the introduction, our approach is inherently tied to handle total orders – which is why, in the aforementioned setup, our solutions will not interleave vertices from different chains  $D_j$ , but rather treat the chains  $D_j$  one after another. Still, our approach allows to do better than simply constructing OTSP solutions for all subinstances  $(G, c, D_j)$  in a black-box way and concatenating them with appropriate shortcutting. The latter would lead to an immediate  $(3/2 + 1/e)\ell$ -approximate solution by using Algorithm 1 on each subinstance. Instead, we observe that after solving the OTSP LP relaxation and sampling trees for each subinstance as in Algorithm 1, we may join all edges obtained this way and only *once* need to connect remaining singletons and do parity correction. This leads to Algorithm 2 as stated below.

Note that, deviating from the above outline, Algorithm 2 starts by guessing a root node  $d_0$  among the minimal nodes in all sets  $D_j$  with respect to  $\prec$ ; this node is used as a common anchor of the given partial orders and results in connectivity of the multigraph containing all sampled trees. To be able to compare the obtained solution to an optimal solution, we need  $d_0$  to be, among the minimal nodes in all sets  $D_j$ , the first one to appear on an optimal solution. We remark that for one  $j \in \{1, \dots, \ell\}$ , we already have  $d_0 \in D_j$ . For the sake of uniform notation, we still add a copy of  $d_0$  to  $D_j$  in Line 2 of Algorithm 2.

■ **Algorithm 2** Approximating a special case of TSP-PC.

---

**Input:** TSP-PC instance  $(G, c, \prec)$  on graph  $G = (V, E)$ , where  $\prec$  precisely induces total orders on disjoint subsets  $D_1, \dots, D_\ell \subseteq V$ .

- 1 Guess a root node  $d_0$  among the minimal nodes in  $D_i$  with respect to  $\prec$ .
- 2 **foreach**  $j \in \{1, \dots, \ell\}$  **do**
- 3     Compute an optimal solution  $(x^{ji}, y^{ji})_{i \in \{0, 1, \dots, |D_j|\}}$  to the OTSP LP relaxation for the OTSP instance  $(G, c, \{d_0\} \dot{\cup} D_j)$  with an order given by  $\prec$  extended by  $d_0 \prec D_j$ .
- 4     **foreach**  $i \in \{0, 1, \dots, |D_j|\}$  **do**
- 5         Apply Lemma 5 to decompose  $(x^{ji}, y^{ji})$  into trees  $\mathcal{T}_{ji}$  with weights  $\mu^{ji}$ .
- 6         Sample one tree  $T_{ji}$  from  $\mathcal{T}_{ji}$  with marginals given by  $\mu^{ji}$ .
- 7 Compute a minimum-cost edge set  $F \subseteq E$  such that the multigraph

$$H := \left( V, F \cup \bigcup_{j=1}^{\ell} \bigcup_{i \in \{1, \dots, |D_j|\}} E[T_{ji}] \right)$$

is connected.

- 8 Let  $Q = \text{odd}(H)$  and compute a minimum cost  $Q$ -join  $J$  in  $G$ .
  - 9 **return** Shortest spanning cycle  $C$  in  $G$  (over all guesses of  $d_0$ ) that visits  $d_0, D_1 \setminus \{d_0\}, \dots, D_\ell \setminus \{d_0\}$  in this order (while respecting  $\prec$  in each  $D_i$ ) and is obtained from  $H \dot{\cup} J$  through Lemma 10.
-

## 1:14 A $(3/2 + 1/e)$ -Approximation Algorithm for Ordered TSP

We show that this algorithm gives the guarantee claimed by Theorem 2 in expectation, and that it can be derandomized using the method of conditional expectations in a way analogous to the derandomization of Algorithm 1.

**Proof of Theorem 2.** Let  $c_{\text{OPT}}$  denote the cost of an optimal solution of the given TSP-PC instance. For every  $j \in \{1, \dots, \ell\}$ , note that the value  $c_{\text{LP}}^j$  of the optimal solution  $(x^{j_i}, y^{j_i})_{i \in \{1, \dots, |D_j|\}}$  to the OTSP instance  $(G, c, \{d_0\} \dot{\cup} D_j)$  generated in Line 2 of Algorithm 2 satisfies  $c_{\text{LP}}^j \leq c_{\text{OPT}}$ . For every  $j \in \{1, \dots, \ell\}$ , denote

$$H_j := \left( V, \bigcup_{i=0}^{|D_j|} E[T_{j_i}] \right) .$$

Every such graph is composed of trees from a connecting tree distribution. Hence, by Item i of Lemma 8,  $H_j$  consists of a large connected component that contains a walk with visits at  $d_0$  and all vertices of  $D_j$  in the desired order, and potentially isolated vertices. For all  $v \notin \{d_0\} \cup D_j$ , Item ii of Lemma 8 implies that

$$\mathbb{P}[v \text{ is isolated in } H_j] \leq \frac{1}{e} .$$

Also, observe that

$$\mathbb{E}[c(E[H_j])] = \sum_{i=0}^{|D_j|} \mathbb{E}[c(T_{j_i})] = \sum_{i=0}^{|D_j|} \sum_{T \in \mathcal{T}_{j_i}} \mu_T^{j_i} c(E[T]) = \sum_{i=0}^{|D_j|} c^\top x^{j_i} = c_{\text{LP}}^j \leq c_{\text{OPT}} .$$

Consequently, the multigraph  $H_0 := \bigcup_{j \in \{1, \dots, \ell\}} H_j$  has total edge cost at most  $\ell \cdot c_{\text{OPT}}$ . Furthermore,  $H_0$  has one large connected component that contains a walk with visits at  $d_0, D_1 \setminus \{d_0\}, \dots, D_j \setminus \{d_0\}$  in this order (obtained by concatenating the walks obtained in the graphs  $H_j$  above), i.e., a walk that respects  $\prec$ . Also, because the graphs  $H_1, \dots, H_\ell$  are independent,

$$\mathbb{P}[v \text{ is isolated in } H_0] = \prod_{j=1}^{\ell} \mathbb{P}[v \text{ is isolated in } H_j] \leq \frac{1}{e^\ell} .$$

Hence, by Item i of Lemma 9, the cost of the minimum-cost edge set  $F$  connecting  $H_0$ , as constructed in Line 2 of Algorithm 2, can be bounded as follows:

$$\mathbb{E}[c(F)] \leq \sum_{v \text{ isolated in } H_0} \mathbb{P}[v \text{ is isolated in } H_0] \cdot c(e_v) \leq \frac{1}{e^\ell} \cdot c(T) \leq \frac{1}{e^\ell} \cdot c_{\text{OPT}} .$$

Here, we used that for any  $j \in \{1, \dots, \ell\}$ , we have  $c(T) \leq c_{\text{LP}}^j$  by Item i of Corollary 4, and  $c_{\text{LP}}^j \leq c_{\text{OPT}}$  as mentioned above. Similarly, by Item ii of Corollary 4, we know that the cost of a cheapest odd( $H$ )-join  $J$  in the multigraph  $H = H_0 \cup F$  can be bounded by  $c(E[J]) \leq \frac{1}{2} \cdot c_{\text{LP}}^j$  for any  $j \in \{1, \dots, \ell\}$ , hence  $c(E[J]) \leq \frac{1}{2} \cdot c_{\text{OPT}}$ .

Altogether, we obtain a connected Eulerian multigraph  $H \dot{\cup} J$  together with a walk that has visits at  $d_0, D_1 \setminus \{d_0\}, \dots, D_j \setminus \{d_0\}$  in the order given by  $\prec$ , and

$$\mathbb{E}[c(E[H \dot{\cup} J])] \leq \left( \ell + \frac{1}{2} + \frac{1}{e^\ell} \right) \cdot c_{\text{OPT}} .$$

Thus, by Lemma 10, we can efficiently find a cycle with visits at  $d_0, D_1 \setminus \{d_0\}, \dots, D_j \setminus \{d_0\}$  in the order given by  $\prec$  of at most the above expected cost.



Finally, to derandomize the random selection of trees  $T_{ji}$  in Algorithm 2, we observe that the present randomized analysis relies on a bound of the form

$$c(C) \leq \sum_{j=1}^{\ell} \sum_{i=0}^{|D_j|} c(T_{ji}) + \sum_{v \notin \dot{\bigcup}_{j \in \{1, \dots, \ell\}} \dot{\bigcup}_{i \in \{1, \dots, |D_j|\}} V[T_{ji}]} c(e_v) + \frac{1}{2} \cdot c_{\text{OPT}} .$$

The conditional expectations of this bound with respect to fixing any subset of the trees  $T_{ji}$  can be readily computed. Thus, the derandomization works analogously to Algorithm 1 by the method of conditional expectations, in each iteration fixing one of the  $T_{ji}$ . To complete the proof of Theorem 6, we observe that all steps of Algorithm 2 can be implemented to run in polynomial time.  $\blacktriangleleft$

► **Remark 11.** We remark that the analysis of Algorithm 2 above is with respect to the actual cost  $c_{\text{OPT}}$  of an optimal TSP-PC solution. Alternatively, after guessing a root node  $d_0$ , one could also write an LP relaxation generalizing the OTSP LP relaxation by introducing independent copies of the variables for each chain  $\{d_0\} \cup D_j$  and minimizing the cost of a point  $x \in P_{\text{HK}}(G)$  that dominates the edge usage within each of the copies. For the ease of presentation, though, we decided to present the above analysis only.

## 4 Proof of Lemma 5

As mentioned earlier, we derive Lemma 5 from a closely related result used by Blauth and Nägele [5, Lemma 4.2]. We restate their result here in a slightly simplified form that follows immediately from the original formulation.

► **Lemma 12** ([5, Lemma 4.2]). *Let  $G = (V, E)$  be a graph with  $r \in V$ , let  $(x, y) \in \mathbb{R}_{\geq 0}^E \times \mathbb{R}_{\geq 0}^V$  be feasible for the system*

$$\begin{aligned} x(\delta(v)) &= 2y_v \quad \forall v \in V \\ x(\delta(S)) &\geq 2y_v \quad \forall S \subseteq V \setminus \{r\}, v \in S \\ y_r &= 1 \end{aligned} \tag{4}$$

and assume that there is a vertex  $u \in V \setminus \{r\}$  such that  $y_u = 1$  and  $e_0 = \{u, r\}$  satisfies  $x_{e_0} \geq 1$ . We can in polynomial time construct a set  $\mathcal{T}$  of trees that all contain the vertices  $r$  and  $u$ , and weights  $\mu \in [0, 1]^T$  with  $\sum_{T \in \mathcal{T}} \mu_T = 1$  and the following properties.

(i) *The point  $x \in \mathbb{R}_{\geq 0}^E$  is a conic combination of the trees in  $\mathcal{T}$  with weights  $\mu$  and the edge  $e_0$ , i.e.,*

$$x = \sum_{T \in \mathcal{T}} \mu_T \chi^{E[T]} + \chi^{e_0} .$$

(ii) *For every  $v \in V \setminus U$ ,*

$$\sum_{T \in \mathcal{T}: v \in V[T]} \mu_T = y_v .$$

The proof of Lemma 12 relies on the well-known *splitting-off* technique (see, e.g., [16,30,31]) applied in the graph  $G$  with weights  $x$ . Indeed, the constraints in the system (4) can be interpreted as  $r$ - $v$  connectivity requirements for all  $v \in V \setminus \{r\}$ , hence splitting-off allows to remove a vertex from the graph while preserving the connectivity properties of the remaining graph. An inductive construction of the desired family of trees is then achieved by reverting the splitting-off operations and extending trees appropriately. For a complete proof, we refer to Blauth and Nägele [5].

To deduce Lemma 5 from Lemma 12, we note that a point  $(x, y) \in P_{s-t \text{ stroll}}$  can be easily transformed into a point  $(x', y')$  satisfying the assumptions of Lemma 12 by adding one unit to  $x_{\{s,t\}}$  and adjusting  $y_s$  and  $y_t$  accordingly. Note that intuitively, this corresponds to closing an  $s$ - $t$  stroll to obtain a tour by adding a copy of the edge  $\{s, t\}$ .

**Proof of Lemma 5.** Given  $(x, y) \in P_{s-t \text{ stroll}}$ , we assume without loss of generality that  $e_0 := \{s, t\} \in E$  and define  $x' := x + \chi^{\{s,t\}}$  and  $y' = y + \frac{1}{2}(\chi^s + \chi^t)$ . We claim that  $(x', y')$  with  $r = s$  and  $u = t$  satisfy the assumptions of Lemma 12. Indeed,  $y'_s = y'_t = 1$ , and  $x'_{e_0} = x_{e_0} + 1 \geq 1$ . Moreover, for  $v \notin \{s, t\}$ , we have  $x'(\delta(v)) = x(\delta(v)) = 2y_v$ ; for  $v \in \{s, t\}$ , we have  $x'(\delta(v)) = x(\delta(v)) + 1 = 2 = 2y'_v$ , hence the degree constraints in (4) are satisfied. Finally, to verify that the cut constraints of (4) are satisfied, too, let  $S \subseteq V \setminus \{r\}$  and  $v \in S$ . If  $t \notin S$ , then  $x'(\delta(S)) = x(\delta(S)) \geq 2y_v = 2y'_v$  follows from the corresponding constraint of  $P_{s-t \text{ stroll}}$ . If otherwise  $t \in S$ , we know that  $x(\delta(S)) \geq 1$ , hence

$$x'(\delta(S)) = x(\delta(S)) + 1 \geq 2 \geq 2y'_v \text{ ,}$$

where we use that  $y'_v = y_v \leq 1$  is implied by the constraints of  $P_{s-t \text{ stroll}}$  for  $v \in V \setminus \{s, t\}$  (see Footnote 5), and  $y'_s = y'_t = 1$ .

Consequently, by applying Lemma 12 to  $(x', y')$ , we obtain in polynomial time a set  $\mathcal{T}$  of trees that all contain  $s$  and  $t$ , and weights  $\mu \in [0, 1]^{\mathcal{T}}$  with  $\sum_{T \in \mathcal{T}} \mu_T = 1$  such that

$$x + \chi^{e_0} = x' = \sum_{T \in \mathcal{T}} \mu_T \chi^{E[T]} + \chi^{e_0} \text{ ,}$$

i.e.,  $x = \sum_{T \in \mathcal{T}} \mu_T \chi^{E[T]}$ , and, for every  $v \in V \setminus \{s, t\}$ ,

$$\sum_{T \in \mathcal{T} : v \in V[T]} \mu_T = y'_v = y_v \text{ .} \quad \blacktriangleleft$$

---


## References

- 1 Susanne Armbruster, Matthias Mnich, and Martin Nägele. A  $(3/2 + 1/e)$ -approximation algorithm for Ordered TSP, 2024. [arXiv:2405.06244v1](https://arxiv.org/abs/2405.06244v1).
- 2 Egon Balas, Matteo Fischetti, and William R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68(1):241–265, 1995. doi:10.1007/BF01585767.
- 3 Jørgen Bang-Jensen, András Frank, and Bill Jackson. Preserving and increasing local edge connectivity in mixed graphs. *SIAM Journal on Discrete Mathematics*, 8(2):155–178, 1995. doi:10.1137/S0036142993226983.
- 4 Jannis Blauth, Nathan Klein, and Martin Nägele. A better-than-1.6-approximation for prize-collecting TSP. In *Proceedings of the 23rd Conference on Integer Programming and Combinatorial Optimization (IPCO '24)*, pages 28–42, 2024. doi:10.1007/978-3-031-59835-7\_3.
- 5 Jannis Blauth and Martin Nägele. An improved approximation guarantee for prize-collecting TSP. In *Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing (STOC '23)*, pages 1848–1861, 2023. doi:10.1145/3564246.3585159.
- 6 Hans-Joachim Böckenhauer, Juraj Hromkovič, Joachim Kneis, and Joachim Kupke. On the approximation hardness of some generalizations of TSP. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT '06)*, pages 184–195, 2006. doi:10.1007/11785293\_19.
- 7 Hans-Joachim Böckenhauer, Tobias Mömke, and Monika Steinová. Improved approximations for TSP with simple precedence constraints. *Journal of Discrete Algorithms*, 21:32–40, 2013. doi:10.1016/j.jda.2013.04.002.
- 8 Martin Böhm, Zachary Friggstad, Tobias Mömke, and Joachim Spoerhase. Approximating TSP variants using a bridge lemma, 2024. [arXiv:2405.12876](https://arxiv.org/abs/2405.12876).

- 9 Moses Charikar, Rajeev Motwani, Prabhakar Raghavan, and Craig Silverstein. Constrained TSP and low-power computing. In *Proceedings of the 5th International Workshop on Algorithms and Data Structures (WADS '97)*, pages 104–115, 1997. doi:10.1007/3-540-63307-3\_51.
- 10 N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.
- 11 N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3, 2022. doi:10.1007/s43069-021-00101-z.
- 12 G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. doi:10.1287/opre.2.4.393.
- 13 Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, and Yihao Zhang. Scheduling with communication delays via LP hierarchies and clustering. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS '20)*, pages 822–833, 2020. doi:10.1109/FOCS46700.2020.00081.
- 14 Vladimir G. Deineko, Michael Hoffmann, Yoshio Okamoto, and Gerhard J. Woeginger. The traveling salesman problem with few inner points. *Operations Research Letters*, 34(1):106–110, 2006. doi:10.1016/j.orl.2005.01.002.
- 15 Jianzhong Du, Joseph Y-T. Leung, and Gilbert H. Young. Scheduling chain-structured tasks to minimize makespan and mean flow time. *Information and Computation*, 92(2):219–236, 1991. doi:10.1016/0890-5401(91)90009-Q.
- 16 András Frank. On a theorem of Mader. *Discrete Mathematics*, 101(1):49–57, 1992. doi:10.1016/0012-365X(92)90589-8.
- 17 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, first edition edition, 1979.
- 18 L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. *Networks*, 48(2):77–89, 2006. doi:10.1002/net.20122.
- 19 R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969. doi:10.1137/0117039.
- 20 Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970. doi:10.1287/opre.18.6.1138.
- 21 Klaus Jansen and Roberto Solis-Oba. Approximation schemes for scheduling jobs with chain precedence constraints. *International Journal of Foundations of Computer Science*, 21(01):27–49, 2010. doi:10.1142/S0129054110007118.
- 22 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21)*, pages 32–45, 2021. doi:10.1145/3406325.3451009.
- 23 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A deterministic better-than-3/2 approximation algorithm for metric TSP. In *Proceedings of the 22nd Conference on Integer Programming and Combinatorial Optimization (IPCO '23)*, pages 261–274, 2023. doi:10.1007/978-3-031-32726-1\_19.
- 24 Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8):1665–1677, 2015. doi:10.1016/j.jcss.2015.06.003.
- 25 Daniil Khachai, Ruslan Sadykov, Olga Battaia, and Michael Khachay. Precedence constrained generalized traveling salesman problem: Polyhedral study, formulations, and branch-and-cut algorithm. *European Journal of Operational Research*, 309(2):488–505, 2023. doi:10.1016/j.ejor.2023.01.039.
- 26 Manfred Kunde. Nonpreemptive LP-scheduling on homogeneous multiprocessor systems. *SIAM Journal on Computing*, 10(1):151–173, 1981. doi:10.1137/0210012.

- 27 J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity results for scheduling chains on a single machine. *European Journal of Operational Research*, 4(4):270–275, 1980. doi:10.1016/0377-2217(80)90111-3.
- 28 Elaine Levey and Thomas Rothvoss. A  $(1 + \epsilon)$ -approximation for makespan scheduling with precedence constraints using LP hierarchies. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC '16)*, pages 168–177, 2016. doi:10.1145/2897518.2897532.
- 29 Mohammad Mehdi Liaee and Hamilton Emmons. Scheduling families of jobs with setup times. *International Journal of Production Economics*, 51(3):165–176, 1997. doi:10.1016/S0925-5273(96)00105-3.
- 30 L. Lovász. On some connectivity properties of Eulerian graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 28(1):129–138, 1976. doi:10.1007/BF01902503.
- 31 W. Mader. A reduction method for edge-connectivity in graphs. *Annals of Discrete Mathematics*, 3:145–164, 1978. doi:10.1016/S0167-5060(08)70504-1.
- 32 Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993. doi:10.1287/moor.18.1.1.
- 33 Yaroslav Saliı. Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization. *European Journal of Operational Research*, 272(1):32–42, 2019. doi:10.1016/j.ejor.2018.06.003.
- 34 Sophia Saller, Jana Koehler, and Andreas Karrenbauer. A systematic review of approximability results for traveling salesman problems leveraging the TSP-T3CO definition scheme, 2023. arXiv:2311.00604.
- 35 A. I. Serdyukov. O nekotorykh ekstremal'nykh obkhodakh v grafakh. *Upravlyaemye sistemy*, 17:76–79, 1987. URL: [http://nas1.math.nsc.ru/aim/journals/us/us17/us17\\_007.pdf](http://nas1.math.nsc.ru/aim/journals/us/us17/us17_007.pdf).
- 36 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the 42nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '10)*, pages 745–754, 2010. doi:10.1145/1806689.1806791.
- 37 Gerhard J. Woeginger. A comment on scheduling on uniform machines under chain-type precedence constraints. *Operations Research Letters*, 26(3):107–109, 2000. doi:10.1016/S0167-6377(99)00076-0.
- 38 Laurence A. Wolsey. *Heuristic analysis, linear programming and branch and bound*, pages 121–134. Springer Berlin Heidelberg, 1980. doi:10.1007/BFb0120913.

# Online Time-Windows TSP with Predictions

Shuchi Chawla ✉ 

University of Texas at Austin, United States

Dimitris Christou ✉ 

University of Texas at Austin, United States

---

## Abstract

In the *Time-Windows TSP* (TW-TSP) we are given requests at different locations on a network; each request is endowed with a reward and an interval of time; the goal is to find a tour that visits as much reward as possible during the corresponding time window. For the online version of this problem, where each request is revealed at the start of its time window, no finite competitive ratio can be obtained. We consider a version of the problem where the algorithm is presented with predictions of where and when the online requests will appear, without any knowledge of the quality of this side information.

Vehicle routing problems such as the TW-TSP can be very sensitive to errors or changes in the input due to the hard time-window constraints, and it is unclear whether imperfect predictions can be used to obtain a finite competitive ratio. We show that good performance can be achieved by explicitly building slack into the solution. Our main result is an online algorithm that achieves a competitive ratio logarithmic in the diameter of the underlying network, matching the performance of the best offline algorithm to within factors that depend on the quality of the provided predictions. The competitive ratio degrades smoothly as a function of the quality and we show that this dependence is tight within constant factors.

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** Travelling Salesman Problem, Predictions, Learning-Augmented Algorithms, Approximation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.2

**Category** APPROX

**Related Version** Due to space limitations, some proofs are deferred to the full version of this paper.

*Full Version:* <https://arxiv.org/abs/2304.01958> [13]

**Funding** *Shuchi Chawla:* This work was funded in part by NSF award CCF-2217069.

## 1 Introduction

Many optimization problems exhibit a large gap in how well they can be optimized offline versus when their input arrives in online fashion. In order to obtain meaningful algorithmic results in the online setting, a natural direction of investigation is to consider “beyond worst case” models that either limit the power of the adversary or increase the power of the algorithm. A recent line of work in this direction has considered the use of *predictions* in bridging the offline versus online gap. Predictions in this context are simply side information about the input that an online algorithm can use for its decision making; the true input is still adversarially chosen and arrives online. The goal is to show that on the one hand, if the predictions are aligned with the input, the algorithm performs nearly as well as in the offline setting (a property known as *consistency*); and on the other hand, if the predictions are completely unrelated to the input, the algorithm nevertheless performs nearly as well as the best online algorithm (a.k.a. *robustness*). *Put simply, good predictions should help, but bad predictions should not hurt, and ideally we should reap the benefits without any upfront knowledge about the quality of the predictions.*



© Shuchi Chawla and Dimitris Christou;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 2; pp. 2:1–2:21



Leibniz International Proceedings in Informatics

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

Predictions have been shown to effectively bypass lower bounds for a variety of different online decision-making problems including, for example, caching [30, 31, 25], scheduling [12, 24, 3], online graph algorithms [4], load balancing [27, 28], online set cover [6], matching problems [17],  $k$ -means [19], secretary problems [1, 18], network design [20, 33, 9] and more.<sup>1</sup>

In this paper, we consider a problem whose objective function value is highly sensitive to changes in the input, presenting a significant challenge for the predictions setting. In the Traveling Salesman Problem with Time Windows (TW-TSP for short), we are given a sequence of service requests at different locations on a weighted undirected graph. Each request is endowed with a reward as well as a time window within which it should be serviced. The goal of the algorithm is to produce a path that maximizes the total reward of the requests visited within their respective time windows. In the online setting, the requests arrive one at a time at the start of their respective time windows, and the algorithm must construct a path incrementally without knowing the locations or time windows of future requests.

Vehicle routing problems such as the TW-TSP that involve hard constraints on the lengths of subpaths (e.g. the time at which a location is visited) are generally more challenging than their length-minimization counterparts. In particular, a small bad decision at the beginning of the algorithm, such as taking a slightly suboptimal path to the first request, can completely obliterate the performance of the algorithm by forcing it to miss out on all future reward. In the offline setting, this means that the approximation algorithm has to carefully counter any routing inefficiency in some segments by intentionally skipping reachable value in other segments. In the online setting, this means that no sublinear competitive ratio is possible.

*Given the sensitivity of the TW-TSP objective to small routing inefficiencies, is it possible to design meaningful online algorithms for this problem using imperfect predictions?*

We consider a model where the algorithm is provided with a predicted sequence of requests at the beginning, each equipped with a predicted location and a predicted time window. The true sequence of requests is revealed over time as before. Of course if the predicted sequence is identical to the true request sequence, the algorithm can match the performance of the best offline algorithm. But what if the predictions are slightly off? Could these small errors cause large losses for the online algorithm? Can the algorithm tolerate large deviations?

Our main result is an online algorithm for the TW-TSP based on predictions whose performance degrades smoothly as a function of the errors in prediction. We obtain this result by explicitly building slack into our solution and benchmark. In a slight departure from previous work on TW-TSP, we require the server to spend one unit of idle “service time” at each served request. We show that this is necessary to obtain a sublinear approximation even with predictions (Theorem 9). (However, in the absence of predictions, the setting with service times continues to admit a linear lower bound on the competitive ratio; See Theorem 8.) We then use service times judiciously in planning a route and accounting for delays caused by prediction errors.

There are two primary sources of error in predictions: (1) the predicted locations of requests may be far from the true locations; and, (2) the predicted time windows may be different from the true time windows. The competitive ratio of our algorithm depends linearly on each of these components, taking the maximum error over each predicted request and normalizing appropriately.<sup>2</sup> This dependence is tight to within constant factors. Besides this

---

<sup>1</sup> A comprehensive compendium of literature on the topic can be found at [29].

<sup>2</sup> Formally, the competitive ratio depends linearly on the ratio of the maximum location error of any predicted request to the minimum service time, as well as the ratio of the maximum time window error to the minimum time window length.

dependence on the prediction error, the competitive ratio depends logarithmically on the diameter of the underlying network, matching the performance of the best known offline algorithm for TW-TSP.

Although our competitive ratio is stated in terms of the maximum location or time window errors, where the maximum is taken over all requests in the instance, our algorithm performs well even when some of the errors are large and most errors are small. In particular, our algorithm’s performance is *simultaneously* competitive against the maximum achievable reward over *any* subset of requests, scaled down by the maximum prediction error over that subset. (See “Extensions” in Section 3 for a formal statement.) In this respect, our guarantees fall within the framework of *metric error with outliers* proposed by [4]. On the other hand, when all or most requests are predicted poorly, our algorithm also inevitably performs poorly as it inherits lower bounds from fully online instances.

Importantly, our algorithm requires little to no information about how the predictions match up against the true requests. For the purpose of analysis, we measure the error in predictions with respect to some underlying matching between the predicted and true requests – the error parameters are then defined in terms of the maximum mismatch between any predicted request and its matched true request. This matching is never revealed to the algorithm and in fact the performance of the algorithm depends on the quality of the *best* possible matching between the predicted and true requests. The only information the algorithm requires about the quality of the predictions is the location error – the maximum distance between any prediction and its matched true request. Even for this parameter, an upper bound suffices (and at a small further loss, a guess suffices).

Our overall approach has several components. The first of these is to construct an instance of the TW-TSP over predicted requests that requires the server to spend some idle time at each request as a “service delay”. We then extend offline TW-TSP algorithms to this service delay setting, obtaining a logarithmic in diameter approximation. We then follow and adapt this offline solution in the online setting. Every time the offline solution services a predicted request, we match this request to a previously revealed true request, take a detour from the computed path to visit and service the true request, and then resume the precomputed path. Altogether this provides the desired competitive ratio.

Our results further generalize to a setting where predictions are coarse in that each single predicted location captures multiple potential true requests that are nearby. We show that with prediction errors defined appropriately, we can again achieve a competitive ratio for this “many to one matching” setting that is logarithmic in the diameter of the graph and polynomial in the prediction error.

Finally, our algorithm and analysis incorporates error in estimating rewards of requests in a straightforward manner achieving the optimal dependence on this third source of error.

## Further related work

Using predictions in the context of online algorithm design was first proposed by [30] for the well-studied caching problem. Since that work, the literature on online algorithm design with predictions has grown rapidly. We point the interested reader to a compendium at [29] for further references.

## Metric error with outliers

Azar et al. [4] initiated the study of predictions in the context of online graph optimization problems, and proposed a framework for quantifying errors in predictions, called *metric error with outliers*, that we adapt. The idea behind this framework is to capture two sources of

error: (1) Some true requests may not be captured by predictions and some predictions may not correspond to any true requests; (2) For requests that are captured by predictions, the predictions may not be fully faithful or accurate. The key observation is that it is possible to design algorithms with performance that depends on these two sources of error *without explicit upfront knowledge of the (partial) correspondence between predicted and true requests*.

In this work, we focus mostly on the second source of error, which we further subdivided into three kinds of error in order to obtain a finer understanding of the relationship between the competitive ratio and different kinds of error. As in the work of Azar et al. [4], we assume that the correspondence between predicted and true requests is never explicitly revealed to the algorithm. The performance of the algorithm nevertheless depends on the error of the *best* matching between predicted and true requests. In Section 3 we describe how the first source of error in Azar et al.’s framework can also be incorporated into our bounds.

### TSP with predictions

Recently a few papers [10, 23, 22] have considered the online TSP and related routing problems with predictions. The input to the online TSP is similar to ours: requests arrive over time in a graph, and a tour must visit each request after its arrival time. However, the objective is different. In our setting, requests also have deadlines, and the algorithm cannot necessarily visit all requests. The goal therefore is to visit as many as possible. In the online TSP, there are no deadlines, and so the objective is to visit *all* requests as quickly as possible, or in other words to minimize the makespan. This makespan minimization objective is typically much easier than the deadline setting, as evidenced by constant factor approximations for it in the offline, online, and predictions settings, as opposed to logarithmic or worse approximations for the latter problem.

The algorithmic idea of precomputing an offline path based on the predictions and then adapting it to the online input has also been used in [10, 23]. The main challenge in the setting that our works considers, is that due to the existence of deadlines, our algorithm needs to be careful on how it adapts its path, as taking a large detour could result in entirely missing the time-windows of future (unrevealed) requests. We circumvent this issue by introducing appropriately large idle times on the predicted requests that our offline solution visits.

### TW-TSP *without* predictions

The (offline) TW-TSP problem has a rich literature and has been studied for over 20 years. The problem is known to be NP-hard even for special cases, e.g. on the line [32], and when all requests have the same release times and deadlines (a.k.a. Orienteering) [11]. Orienteering admits constant factor approximations [11, 7, 14], and even a PTAS when requests lie in a fixed dimensional Euclidean space [2, 16]. For general time windows, constant factor approximations are only known for certain special cases: e.g. constant number of distinct time windows [15]; and on line graphs [32, 26, 8, 21]. For general graphs and time-windows, the best approximations known are logarithmic in input parameters [7, 14].

To the best of our knowledge, the online setting for TW-TSP has only been considered by Azar and Vardi [5]. Azar and Vardi assume that service times are non-zero and present competitive algorithms under the assumption that the smallest time window length  $L_{\min}$  is comparable to the diameter  $D$  of the graph, as no sublinear competitive ratio can be achieved if  $L_{\min} < D/2$ . We are able to beat this lower bound by relying on predictions.



## Organization of the paper

We formally define the problem and our error model in Section 2. Section 3 describes our results and provides an outline for our analysis. All of our main results are covered in that section. Proofs of these results can be found in subsequent sections. In particular, Sections 4, 5, and 6 fill in the details of our upper bound, and Section 7 proves the stated lower bounds. Proofs omitted from the main body of this paper can be found in the appendix of the full version [13].

## 2 Definitions

### 2.1 The Traveling Salesman Problem with Time-Windows

An instance of the TW-TSP consists of a network  $G$  and a (finite) sequence of service requests  $I$ . Here,  $G = (V, E, \ell)$  is an undirected network with edge lengths  $\{\ell_e\}_{e \in E}$ . Extending the notion of distance to all vertex pairs in  $G$ , we define  $\ell(u, v)$  for  $u, v \in V$  to be the length of the shortest path from  $u$  to  $v$ . We assume without loss of generality that  $G$  is connected and that the edge lengths  $\ell_e$  are integers. A service request  $\sigma = (v_\sigma, r_\sigma, d_\sigma, \pi_\sigma)$  consists of a vertex  $v_\sigma \in V$  at which the request arrives, a release time  $r_\sigma \in \mathbb{Z}^+$ , a deadline  $d_\sigma \in \mathbb{Z}^+$  with  $d_\sigma > r_\sigma$ , and a reward  $\pi_\sigma \in \mathbb{Z}^+$ . We use  $\Sigma \subseteq V \times \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}^+$  to denote the set of all possible client requests and  $I \subset \Sigma$  to denote the set of requests received by the algorithm.

The solution to TW-TSP is a continuous *walk* on  $G$  that is allowed to remain idle on the vertices of the graph.<sup>3</sup> Formally, the walk starts from some vertex at time  $t = 0$ ; at every time-step that it occupies a vertex  $u \in V$ , it can either remain idle on  $u$  for some number of time-steps or it can move to some  $v \in V$  by spending time  $t = \ell(u, v)$ ; we comment that while the path is mid-transition, no more decisions can be made. Notice that this creates a notion of a discrete time-horizon that will be important towards formalizing the online variant of the problem.

We use  $\mathcal{W}(G)$  to denote the set of all walks on  $G$ . Given a request  $\sigma \in \Sigma$ , we say that a walk  $W$  *covers* it if  $W$  remains idle on vertex  $v_\sigma$  for at least one time-step,<sup>4</sup> starting on some step  $\tau \in [r_\sigma, d_\sigma - 1]$ . For a sequence of requests  $I \subset \Sigma$ , we use  $\text{Cov}(W, I) \subseteq I$  to denote the set of requests in  $I$  that are covered by  $W$ . Then, the reward obtained by walk  $W$  is denoted by  $\text{Rew}(W, I) := \sum_{\sigma \in \text{Cov}(W, I)} \pi_\sigma$ . The objective of TW-TSP is to compute a walk  $W \in \mathcal{W}(G)$  of maximum reward. We denote this by  $\text{OPT}(G, I) := \max_{W \in \mathcal{W}(G)} [\text{Rew}(W, I)]$ .

### 2.2 The offline, online, and predictions settings

We assume that the network  $G$  is known to the algorithm upfront in all of the settings we consider. In the **offline** version of the problem, the sequence of requests  $I$  is given to the algorithm in advance. In the **online** version, requests  $\sigma \in I$  arrive in an online fashion; specifically, each request  $\sigma \in I$  is revealed to the algorithm at its release time  $r_\sigma$ .

In the **predictions** setting, the true sequence of requests  $I$  arrives online, as in the online setting. However, the algorithm is also provided with a predicted sequence  $I' \subset \Sigma$  in advance, where every request  $\sigma' \in I'$  is endowed with a location, a time window, and a reward. The

<sup>3</sup> To keep our exposition simple, we do not specify a starting location for the walk. However, all of our algorithms can be adapted without loss to the case where a starting location is fixed, as described towards the end of Section 3.

<sup>4</sup> As we mentioned in the introduction, this requirement of a minimal one-unit service time is necessary in order to achieve any sublinear approximation for the online TW-TSP even with predictions. See Theorem 9 in Section 7.

quality of predictions is expressed in terms of their closeness to true requests. To this end, we define three notions of mismatch or error. For a true request  $\sigma$  and predicted request  $\sigma'$ , the location error, time windows error, and reward error are defined as:

$$\begin{aligned}\text{LocErr}(\sigma, \sigma') &:= \ell(v_\sigma, v_{\sigma'}) \\ \text{TWErr}(\sigma, \sigma') &:= \max\{|r_\sigma - r_{\sigma'}|, |d_\sigma - d_{\sigma'}|\} \\ \text{RewErr}(\sigma, \sigma') &:= \max\{\pi_\sigma/\pi_{\sigma'}, \pi_{\sigma'}/\pi_\sigma\}\end{aligned}$$

We extend these definitions to the entire sequences  $I$  and  $I'$  through an underlying (but unknown to the algorithm) matching between the requests in the two lists:

► **Definition 1.** *Given two request sequences  $I, I' \subset \Sigma$  with  $|I| = |I'|$  and a perfect matching  $M : I \mapsto I'$ , we define the location, time window, and reward errors for the matching  $M$  as:*

$$\begin{aligned}\Lambda_M &:= \max_{\sigma \in I} \text{LocErr}(\sigma, M(\sigma)) \\ \tau_M &:= \max_{\sigma \in I} \text{TWErr}(\sigma, M(\sigma)) \\ \rho_M &:= \max_{\sigma \in I} \text{RewErr}(\sigma, M(\sigma))\end{aligned}$$

We use  $n = |V|$  to denote the number of vertices in  $G$ ,  $D$  to denote the diameter of the graph, and  $L_{\min}$  and  $L_{\max}$  to denote the size of the smallest and largest time windows respectively (of a true or predicted request) in the given instance; that is, we denote  $L_{\min} = \min_{\sigma \in I \cup I'} |d_\sigma - r_\sigma|$  and  $L_{\max} = \max_{\sigma \in I \cup I'} |d_\sigma - r_\sigma|$ . The competitive ratios of the algorithms we develop depend on these parameters.

### Knowns and unknowns

We denote an instance of the TW-TSP with predictions by  $(G, I, I', M)$ . All components of the instance are chosen adversarially. As mentioned earlier, the network  $G$  and the predicted sequence  $I'$  are provided to the algorithm at the start. The sequence  $I$  arrives online. We assume that the algorithm receives no direct information about the matching  $M$ , but is provided with an upper bound on the error  $\Lambda_M$ . We will also assume that the algorithm knows the parameter  $L_{\min}$ , although this is without loss of generality as the parameter can be inferred within constant factor accuracy from the predictions.<sup>5</sup>

## 2.3 The TW-TSP with service times

At a high level our algorithm has two components: an offline component that computes a high-reward walk over the predicted locations of requests, and an online component that largely follows this walk but takes “detours” to cover the arriving true sequence of requests. In particular, as the algorithm follows the offline walk, for each predicted location it visits where a “close by” true request is available, the algorithm takes a “detour” to this true request, returns back to the predicted location, and resumes the remainder of the walk. In order to incorporate the time spent taking these detours in our computation of the offline walk, we require the walk to spend some “service time” at each predicted location it covers. Accordingly, we define a generalization of the TW-TSP:

<sup>5</sup> In particular, assuming  $\tau_M \leq L_{\min}/2$ , which is necessary for our results to hold, the time window of any true request can be no shorter than half the smallest time window of any predicted request.

► **Definition 2.** The TW-TSP with Service Times (TW-TSP-S) takes as input a network  $G$ , a sequence of service requests  $I$ , and a service time  $S \in \mathbb{Z}^+$ , and returns a walk  $W \in \mathcal{W}(G)$ . We say that  $W$  covers a request  $\sigma \in I$ , denoted  $\sigma \in \text{Cov}(W, I, S)$ , if it remains idle on vertex  $v_\sigma$  for at least  $S$  time steps, starting at some step  $t \in [r_\sigma, d_\sigma - S]$ . We define the reward of  $W$  as  $\text{Rew}(W, I, S) := \sum_{\sigma \in \text{Cov}(W, I, S)} \pi_\sigma$ . The optimal value of the instance is given by:

$$\text{OPT}(G, I, S) := \max_{W \in \mathcal{W}(G)} [\text{Rew}(W, I, S)].$$

Note that the original version of TW-TSP as defined previously simply corresponds to the special case of TW-TSP-S with service time  $S = 1$ , and in particular, we have  $\text{Rew}(W, I) = \text{Rew}(W, I, 1)$ , and  $\text{OPT}(G, I) = \text{OPT}(G, I, 1)$ .

### 3 Our results and an outline of our approach

Our main result is as follows.

► **Theorem 3.** Given any instance  $(G, I, I', M)$  of the TW-TSP with predictions whose errors satisfy  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq (L_{\min} - 1)/4$ , there exists a polynomial-time online algorithm that takes the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input, and constructs a walk  $W \in \mathcal{W}(G)$  such that

$$\mathbb{E}[\text{Rew}(W, I)] \geq \frac{1}{O(\Lambda_M \cdot \rho_M^2 \cdot \log \min(D, L_{\max}))} \cdot \text{OPT}(G, I).$$

As mentioned previously, our algorithm consists of two components. The offline component constructs a potential walk in the network with the help of the predicted requests. Then an online component adapts this walk to cover true requests that arrive one at a time. We break up the design and analysis of our algorithm into four steps. The first two steps relate the offline instance we solve to the hindsight optimal solution for the online instance. The third step then applies an offline approximation to the predicted instance with appropriate service times. The final step deals with the online adaptation of the walk to the arriving requests.

The following four lemmas capture the four steps. First, we show (Section 4) that introducing a service time of  $S$  hurts the optimal value by at most a factor of  $2S - 1$ . As we prove in Lemma 14 of Section 4, this dependency on  $S$  is tight. Observe that we require  $S \leq L_{\min}$ , as for any tour to feasibly cover a request, the service time for that request must fit within its time window.

► **Lemma 4.** For any instance  $(G, I)$  of the TW-TSP with service times, and any integer  $S \leq L_{\min}$ , we have

$$\text{OPT}(G, I, S) \geq \frac{1}{2S - 1} \cdot \text{OPT}(G, I, 1).$$

Our second step (also in Section 4) relates the value of the optimal solution over the true requests  $I$  to the optimum over the predicted sequence  $I'$ . In both cases, we impose some service time requirements. Note that this argument needs to account for the discrepancy in locations, time windows, as well as the rewards of the true and predicted requests.

► **Lemma 5.** Let  $(G, I, I', M)$  be an instance of the TW-TSP with predictions, where  $\Lambda_M$ ,  $\rho_M$ , and  $\tau_M$  denote the maximum location, reward, and time window errors of the instance respectively. Define  $S := 4\Lambda_M + 1$  and  $S' := 2\Lambda_M + 1$ . Then, if  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq (L_{\min} - 1)/4$ , we have

$$\text{OPT}(G, I', S') \geq \frac{1}{3\rho_M} \cdot \text{OPT}(G, I, S).$$

Our third step (Section 5) captures the offline component of our algorithm: computing an approximately optimal walk over the predicted requests with the specified service times. For this we leverage previous work on the TW-TSP without service times and show how to adapt it to capture the service time requirement.

► **Lemma 6.** *Given any instance  $(G, I', S')$  of the TW-TSP with service times, there exists a polynomial time algorithm that returns a walk  $W \in \mathcal{W}(G)$  with reward*

$$\text{Rew}(W, I', S') = \frac{1}{\mathcal{O}(\log \min(D, L_{\max}))} \cdot \text{OPT}(G, I', S').$$

Finally, the fourth component (Section 6) addresses the online part of our algorithm. Given a walk computed over the predicted request sequence, it solves an appropriate online matching problem to construct detours to capture true requests. As in Lemma 5, this part again needs to account for the discrepancy in locations, time windows, as well as the rewards of the true and predicted requests.

► **Lemma 7.** *Given an instance  $(G, I, I', M)$  of the TW-TSP with predictions satisfying  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq (L_{\min} - 1)/4$ ; a walk  $W' \in \mathcal{W}(G)$ ; and any integer  $S' \geq 2\Lambda_M + 1$ , there exists an online algorithm (Algorithm 1) that returns a walk  $W \in \mathcal{W}(G)$  with expected reward*

$$\mathbb{E}[\text{Rew}(W, I, 1)] \geq \frac{1}{6\rho_M} \cdot \text{Rew}(W', I', S').$$

Theorem 3 follows immediately by putting Lemmas 4, 5, 6 and 7 together.

### Lower bounds and tightness of our results

We show that the online TW-TSP does not admit sublinear competitive algorithms in the absence of predictions if  $L_{\min} < D$ , even with non-zero service times. Furthermore, if the service times are all 0, no sublinear competitive ratio is possible even using predictions that are accurate in all respects except the request location. Therefore, in order to achieve a nontrivial competitive ratio, it is necessary to use predictions as well as to impose non-zero service times on the optimum. The proofs are presented in Section 7.

► **Theorem 8.** *The competitive ratio of any randomized online algorithm for Online TW-TSP on instances with  $L_{\min} \leq D$  and all service times equal to 1 is at most  $1/n$ .*

► **Theorem 9.** *For any  $S > 0$ , there exists an instance  $(G, I, I', M)$  of the TW-TSP with predictions and service times 0, satisfying  $\tau_M = 0$ ,  $\rho_M = 1$ , and  $\Lambda_M = S$ , such that any randomized online algorithm taking the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input achieves a reward no larger than  $O(1/n) \cdot \text{OPT}(G, I, 0)$ . Here  $n$  is the number of vertices in  $G$ .*

As mentioned earlier, the best known approximation factor for the offline TW-TSP is  $O(\log L_{\max})$  (which we show can be improved slightly to  $O(\log \min(D, L_{\max}))$ ). We inherit this logarithmic dependence on  $D$  and  $L_{\max}$  in the predictions setting. Furthermore, any improvements to the offline approximation would immediately carry through into our competitive ratio as well. In particular, given an offline TW-TSP algorithm that achieves a competitive ratio of  $\alpha(D, L_{\max})$ , we obtain an online algorithm that achieves a competitive ratio of  $O(\Lambda_M \cdot \rho_M^2 \cdot \alpha(D, L_{\max}))$ .

The dependence of our bound on  $\rho_M$  can easily be seen to be tight – consider a star graph with requests on leaves, and edge lengths and time windows defined in such a manner that any feasible walk can cover at most one request. Then an uncertainty of a factor of

$\rho_M$  in the predicted rewards can force any online algorithm to obtain an  $\Omega(\rho_M^2)$  competitive ratio even if the predictions are otherwise perfect. Finally, we show in Section 7.2 that the dependence of our competitive ratio on  $\Lambda_M$  is also tight:

► **Theorem 10.** *For any  $S > 0$ , there exists an instance  $(G, I, I', M)$  of the TW-TSP with predictions satisfying  $\tau_M = 0$ ,  $\rho_M = 1$ , and  $\Lambda_M = S$  such that the competitive ratio of any randomized online algorithm taking the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input asymptotically approaches  $1/(S + 1)$ .*

## Extensions and generalizations

We now describe some ways in which we can weaken the assumptions in Theorem 3 while maintaining its competitive ratio guarantee:

- **Lack of knowledge of  $\Lambda_M$ .** Our algorithm continues to work as intended if it is provided with an upper bound on  $\Lambda_M$  rather than the exact value of the parameter, with the performance of the algorithm degrading linearly with the upper bound, as in the theorem above. One such upper bound is simply  $L_{\min}/4$ . Moreover, by guessing  $\Lambda_M$  within a factor of 2 in the range  $[0, L_{\min}/4]$ , we can obtain the claimed approximation with a further loss of  $\mathcal{O}(\log L_{\min})$ . Thus, our algorithm can achieve non-trivial guarantees that scale with the location error even in settings where no information is given about any of the prediction errors  $\Lambda_M, \tau_M, \rho_M$ .
- **Assumptions on  $\tau_M$  and  $\Lambda_M$ .** It is easy to see that it is necessary to assume  $\Lambda_M \leq L_{\min}$  to obtain a nontrivial competitive ratio, as predictions with a location error larger than the time window size are of no value to the online algorithm. On the other hand, assuming  $\tau_M \leq L_{\min}$  is not necessary. We can accommodate larger time window errors by following one out of roughly  $\tau_M/L_{\min}$  different time shifts of the offline walk. This worsens our approximation factor by an additional factor of  $\tau_M/L_{\min}$ . In particular, this algorithm achieves a competitive ratio of  $O(\Lambda_M \cdot \rho_M^2 \cdot \tau_M/L_{\min} \cdot \log \min(D, L_{\max}))$ .
- **Random rewards.** Our results also hold in the case of random rewards. Specifically, consider a setting where the rewards  $\{\pi_\sigma\}_{\sigma \in I}$  are drawn from some joint (not necessarily product) distribution  $D$  over  $\mathbb{R}_+^I$ . In that case, we define  $\text{Rew}(W, I) := \sum_{\sigma \in \text{Cov}(W, I)} \mathbb{E}[\pi_\sigma]$ , and  $\text{OPT}(G, I)$  as the maximum reward obtained by any walk  $W \in \mathcal{W}(G)$ .<sup>6</sup> Finally, we define  $\text{RewErr}(\sigma, \sigma')$  as the mismatch between  $\pi'_\sigma$  and  $\mathbb{E}[\pi_\sigma]$ . Our analysis provides the same approximation as before in this setting. See the full version for a formal proof.

► **Corollary 11.** *Given an instance  $(G, I, I', M)$  of the TW-TSP with predictions where requests have randomly drawn rewards, and predictions errors satisfy that  $\tau_M \leq L_{\min}/2$  and also  $\Lambda_M \leq (L_{\min} - 1)/4$ , there exists a polynomial-time online algorithm that takes the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input, and constructs a walk  $W \in \mathcal{W}(G)$  such that*

$$\mathbb{E}[\text{Rew}(W, I)] \geq \frac{1}{O(\Lambda_M \cdot \rho_M^2 \cdot \log \min(D, L_{\max}))} \cdot \text{OPT}(G, I)$$

<sup>6</sup> Note that we do not allow the optimal walk to adapt to instantiations of rewards. Adaptive walks cannot be competed against in an online setting even with predictions.

- **Routed instances.** Next, we consider the case where a starting vertex  $v_0$  is also specified, and the solution space  $\mathcal{W}(G)$  includes all walks on  $G$  that *start* on vertex  $v_0$  at  $t = 0$ . We can easily see that this setting is essentially equivalent to its unrooted counterpart, under the extra assumption that each request  $\sigma = (v_\sigma, r_\sigma, d_\sigma, \pi_\sigma)$  satisfies the conditions  $\ell(v_0, v_\sigma) \leq r_\sigma$ . This is a reasonable assumption as no algorithm can visit a request  $\sigma$  before time  $\ell(v_0, v_\sigma)$  anyway. Clearly, for any rooted instance  $(G, I, v_0)$ , the unrooted optimal  $\text{OPT}(G, I)$  is an upper bound on the rooted optimal  $\text{OPT}(G, I, v_0)$ . On the other hand, the unrooted path computed by our algorithm can be transformed to a path of same reward rooted at  $v_0$  by going directly from  $v_0$  to the predicted request serviced first, as this distance is at most equal to the request's release time.
- **Partial matching.** Next we consider the case where not all true requests are captured by the predicted requests and, on the flip side, where some predicted requests do not correspond to true requests at all. Following the framework of [4], we consider partial matchings between  $I$  and  $I'$ , and define  $\Delta_1^M$  to be the total reward of all true requests that are unmatched, and  $\Delta_2^M$  to be the total predicted reward of predicted requests that are unmatched. Then, it is easy to see that our analysis goes through for the subsets of  $I$  and  $I'$  that are matched to each other, costing us an additive amount of no more than  $\Delta_1^M + \Delta_2^M$ . See the full version for a formal proof.

► **Corollary 12.** *Given an instance  $(G, I, I')$  of the TW-TSP with predictions, let  $M$  be any (incomplete) matching between  $I$  and  $I'$ , and let the error parameters  $\Lambda_M, \rho_M, \tau_M, \Delta_1^M$ , and  $\Delta_2^M$  be defined as above. Then, there exists an online algorithm that takes  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input, and returns a walk  $W \in \mathcal{W}(G)$  such that*

$$\mathbb{E}[\text{Rew}(W, I)] \geq \Omega\left(\frac{1}{\Lambda_M \cdot \rho_M^2 \cdot \log \min(D, L_{\max})}\right) \cdot (\text{OPT}(G, I) - \Delta_1^M) - \frac{\Delta_2^M}{\rho_M}.$$

- **Many to one matching.** Consider a setting where predictions are coarse in that each single predicted location captures multiple potential true requests. We can model such a setting within our predictions framework and obtain almost the same guarantee as in Theorem 3. In particular, for this setting, let  $M$  be a many-to-one matching from  $I$  to  $I'$ . We define the location error of a predicted request  $\sigma' \in I'$  as the length of the shortest path that starts at  $\sigma'$ , visits all of the locations of the true requests that are preimages of  $\sigma'$  in  $M$ , spending one unit of time at each, and returns back to  $\sigma'$ . Observe that this location error is the length of the optimal solution to an orienteering problem rooted at  $\sigma'$ . Correspondingly, we want the reward associated with  $\sigma'$  to capture the total reward of all the true requests matched to  $\sigma'$ , and define its reward error accordingly. Finally, the time window error is defined as before, as a maximum over all pairs  $\sigma$  and  $\sigma'$  that are matched to each other. Our algorithm for the setting of Theorem 3 constructs a matching between  $I'$  and  $I$  in an online fashion. For this one to many setting, we solve instances of the orienteering problem rooted at each predicted request we visit. The performance of the algorithm accordingly worsens by a small constant factor and we achieve a competitive ratio of  $O(\Lambda_M \rho_M^2 \log \min(D, L_{\max}))$  as before. Due to space limitations, the details of the proof are omitted from this version. See Section 8 of the full version [13] for further details.

► **Theorem 13.** *Given an instance  $(G, I, I', M)$  of the TW-TSP with predictions where  $M$  is a many-to-one matching with errors as defined above, and satisfying  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq L_{\min}/2$ , there exists a polynomial-time online algorithm that takes the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input, and constructs a walk  $W \in \mathcal{W}(G)$  such that*

$$\mathbb{E}[\text{Rew}(W, I)] \geq \frac{1}{O(\Lambda_M \cdot \rho_M^2 \cdot \log \min(D, L_{\max}))} \cdot \text{OPT}(G, I).$$

#### 4 Relating the Optima

In this section we provide the proofs of Lemmas 4 and 5 that relate the optima over the true and the predicted request sequences, using service times as a mechanism to capture the prediction errors. We begin by proving that a service time of  $S$  can hurt the optimal by at most a factor of  $2S - 1$ .

► **Lemma 4.** *For any instance  $(G, I)$  of the TW-TSP with service times, and any integer  $S \leq L_{\min}$ , we have*

$$\text{OPT}(G, I, S) \geq \frac{1}{2S - 1} \cdot \text{OPT}(G, I, 1).$$

**Proof.** Let  $W \in \mathcal{W}(G)$  be the walk that achieves the optimum  $\text{OPT}(G, I, 1)$ , and let the requests in  $I$  that are covered by  $W$  be denoted as  $\sigma_i = (v_i, r_i, d_i, \pi_i)$  and ordered in the sequence in which they are covered by  $W$ . The lemma follows directly from the simple observation that if we don't service the  $(S - 1)$ -requests *prior* and *after* some request  $\sigma_i$ , then we can save enough time to service  $\sigma_i$  for  $S$  time-steps within its time window.

Formally, if  $t_i \in [r_i, d_i - 1]$  is the step at which  $W$  begins servicing request  $\sigma_i$ , then by skipping the idle times on the  $(S - 1)$ -previous and next requests we can remain idle on  $v_i$  from step  $t_i - (S - 1)$  until step  $t_i + S$  (since  $W$  already remained idle on  $v_i$  for 1 step) while still being able to keep up with walk  $W$ . Since  $S \leq L_{\min}$ , it is easy to verify that at least  $S$  of these time-steps are going to fall in the time-window  $[r_i, d_i]$ .

We now partition the requests  $\sigma_i = (v_i, r_i, d_i, \pi_i)$  into  $2S - 1$  sub-sequences, each of which starts at some request  $i \in [S]$ , and covers the requests  $\sigma_i, \sigma_{i+(2S-1)}, \sigma_{i+2(2S-1)}$ , and so forth. Each such sequence can be covered with a walk, with idle times built in as above, so as to be feasible for the instance  $(G, I, S)$ . Clearly, one of these walks obtains a reward of at least  $\text{OPT}(G, I, 1)/(2S - 1)$ , completing the proof. ◀

In the appendix of the full version, we show that the above lemma obtains a tight gap between the optima at different service times.

► **Lemma 14.** *For any pair of integers  $(L, S)$  such that  $L \geq 2S - 2 \geq 1$ , there exists a rooted instance  $(G, I)$  of the TW-TSP with service costs such that  $L_{\min} = L$  and*

$$\text{OPT}(G, I, S) = \frac{1}{2S - 1} \cdot \text{OPT}(G, I, 1).$$

Next, we provide the proof of Lemma 5 that relates the optima between the predicted and true request sequences, by appropriately addressing all three possible types of prediction errors.

► **Lemma 5.** *Let  $(G, I, I', M)$  be an instance of the TW-TSP with predictions, where  $\Lambda_M$ ,  $\rho_M$ , and  $\tau_M$  denote the maximum location, reward, and time window errors of the instance respectively. Define  $S := 4\Lambda_M + 1$  and  $S' := 2\Lambda_M + 1$ . Then, if  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq (L_{\min} - 1)/4$ , we have*

$$\text{OPT}(G, I', S') \geq \frac{1}{3\rho_M} \cdot \text{OPT}(G, I, S).$$

**Proof.** Let  $W$  be the walk that achieves the optimum  $\text{OPT}(G, I, S)$ , and let the requests in  $I$  covered by  $W$  be denoted as  $\sigma_i = (v_i, r_i, d_i, \pi_i)$  and ordered in the sequence in which they are visited by  $W$ . Let  $\sigma'_i = (v'_i, r'_i, d'_i, \pi'_i)$  denote the predicted request matched to  $\sigma_i$ , that is,  $\sigma'_i = M(\sigma_i)$ . Observe that the total reward of all requests  $\{\sigma'_i\}$  corresponding to  $\sigma_i \in \text{Cov}(W, I, S)$  is at least  $\text{Rew}(W, I, S)/\rho_M$ .

We will consider a walk  $W'$  in  $G$  defined as follows. The walk  $W'$  follows  $W$ , visiting the requests  $\sigma_i$  in sequence. As soon as  $W$  starts servicing  $\sigma_i$ ,  $W'$  takes a detour to visit  $\sigma'_i$ ; remains idle at  $\sigma'_i$  for  $S'$  time steps; returns back to  $\sigma_i$ ; remains idle at  $\sigma_i$  for  $S - 2\ell(v_i, v'_i) - S' \geq 0$  time steps; and then resumes the walk  $W$ . Observe that  $W'$  is identical to  $W$  outside of the detours it takes to visit the  $\sigma'_i$ 's.

Our goal is to feasibly capture all of the reward contained in the  $\sigma'_i$ 's. The problem is that the walk  $W'$  may miss some of this reward due to the mismatch in the time windows of the true and predicted requests. To this end, we will consider two variations of the walk  $W'$ . Let  $K := L_{\min}/2 \geq \tau_M$ . The walk  $W'_1$  is identical to  $W'$  except that it starts  $K$  steps after  $W'$  starts, and accordingly visits every location exactly  $K$  steps after  $W'$  visits it. The walk  $W'_2$  is identical to  $W'$  except that it starts  $K$  steps *before*  $W'$  starts,<sup>7</sup> and accordingly visits every location exactly  $K$  steps before  $W'$  visits it.

Now consider some  $\sigma'_i$  corresponding to a request  $\sigma_i$  covered by  $W$  in the instance  $(G, I, S)$ . We claim that at least one of the walks  $W'$ ,  $W'_1$ , and  $W'_2$  covers  $\sigma'_i$  in  $(G, I', S')$ . Let  $t$  be the time at which  $W'$  arrives at  $v'_i$ ; recall that  $W'$  remains at the node until at least  $t + S'$ . Note that  $t \geq r_i$  and  $t + S' \leq d_i$  due to  $\sigma_i \in \text{Cov}(W, I, S)$ .

First, suppose that  $r'_i \leq t$  and  $d'_i \geq t + S'$ , then  $\sigma'_i$  is covered by  $W'$  in  $(G, I', S')$ . Next suppose that  $r'_i > t$ . Then,  $W'_1$  arrives at  $v'_i$  at time  $t + K \geq r_i + K \geq r_i + \tau_M \geq r'_i$ . On the other hand, it remains at  $v'_i$  until time  $t + K + S' < r'_i + K + S' \leq r'_i + L_{\min} \leq d'_i$ . Therefore,  $\sigma'_i$  is covered by  $W'_1$ . Finally, suppose that  $d'_i < t + S'$ . Then,  $W'_2$  arrives at  $v'_i$  at time  $t - K > d'_i - S' - K \geq d'_i - L_{\min} \geq r'_i$ . On the other hand, it remains at  $v'_i$  until time  $t - K + S' \leq d_i - K \leq d_i - \tau_M \leq d'_i$ . Therefore,  $\sigma'_i$  is covered by  $W'_2$ .

We get that at least one of  $W'$ ,  $W'_1$ , or  $W'_2$  obtains at least a  $1/3\rho_M$  fraction of  $\text{OPT}(G, I, S)$ , where the factor of  $\rho_M$  is lost due to the mismatch in the predicted rewards. The lemma follows directly from this.  $\blacktriangleleft$

## 5 The offline approximation

In this section, we design an  $O(\log \min(D, L_{\max}))$  deterministic and polynomial-time approximation algorithm for the TW-TSP with service times, providing the proof of Lemma 6. Our proof relies on a series of reductions between different offline problems, applications of existing algorithms as well as the design of novel algorithmic components. We break up our argument into a series of lemmas. Due to space limitations, all the proofs are moved to the appendix of the full version [13].

1. First, we designing an  $O(\log \min(D, L_{\max}))$  approximation algorithm for TW-TSP (without service times). Since the work of [14] already provides a  $O(\log L_{\max})$  approximation for the setting with integer time-windows (see Lemma 5.3 of [14]), it suffices to prove the following:

► **Lemma 15.** *Given an instance of the TW-TSP (without service times) with  $L_{\min} \geq 4D$ , there exists a polynomial time algorithm that achieves an  $O(1)$  approximation.*

Our proof relies on the observation that when time-windows are sufficiently large compared to the diameter of the graph, the problem essentially reduces to an instance of the well-studied Orienteering problem, for which constant approximation algorithms are

<sup>7</sup> To be precise, this walk starts at the location where  $W'$  is at at step  $K$ .



known. We comment that similar ideas have been used in [5]. Then, it is straightforward to combine this algorithm together with the algorithm of [14] to acquire an  $O(\log \min(L_{\max}, D))$  approximation of TW-TSP.

► **Lemma 16.** *There exists an  $O(\log \min(D, L_{\max}))$  approximation algorithm for the TW-TSP problem.*

2. Next, we design a simple approximation-preserving reduction from TW-TSP with service times to TW-TSP (without service times). The main idea behind this reduction is to treat service times as edge lengths in an augmented graph whose diameter is roughly  $D + S$ . For instances with  $S \leq D$ , this increase becomes negligible and thus by combining our reduction with Lemma 16, we immediately get the following:

► **Lemma 17.** *Given an instance  $(G, I, S)$  of the TW-TSP with service times such that  $S \leq D$ , there exists a polynomial time algorithm that achieves an  $O(\log \min(D, L_{\max}))$  approximation.*

3. Finally, we handle the case of large service times, specifically  $S \geq D$ . In that case, it turns out that we can reduce the instance to one over a uniform complete graph. Then, the TW-TSP-S essentially becomes equivalent to the well-studied *Job Scheduling* problem, for which constant approximations are known.

► **Lemma 18.** *Given an instance  $(G, I, S)$  of the TW-TSP with service costs such that  $S \leq D$ , there exists a polynomial time algorithm that achieves an  $O(1)$  approximation.*

The proof of Lemma 6 follows immediately from Lemma 17 and Lemma 18. We comment that any improvement in the best known approximation algorithm for TW-TSP will immediately imply an improvement for all the results that this work presents. Lemma 18 essentially enables us to assume that in all instances of interest,  $S \leq D$ . Under this assumption, our reduction used in the proof of Lemma 17 essentially states that TW-TSP-S becomes equivalent to TW-TSP in graphs of diameter  $\mathcal{O}(D)$  and maximum window size  $\mathcal{O}(L_{\max})$ . As an immediate corollary, given an offline TW-TSP algorithm that achieves a competitive ratio of  $\alpha(D, L_{\max})$ , we immediately obtain an offline  $O(\alpha(D, L_{\max}))$  approximation for TW-TSP-S, that can be used in order to substitute Lemma 6 in our analysis and improve the competitive ratio of Theorem 3.

## 6 The online algorithm

In this section we present an online algorithm that takes as input a pre-computed walk over the predicted request sequence and solves an appropriate online matching problem in order to construct detours that capture true requests, while taking into account the possible errors in the predictions. The formal guarantee of our algorithm is given in Lemma 7, which we restate for the reader's convenience:

► **Lemma 7.** *Given an instance  $(G, I, I', M)$  of the TW-TSP with predictions satisfying  $\tau_M \leq L_{\min}/2$  and  $\Lambda_M \leq (L_{\min} - 1)/4$ ; a walk  $W' \in \mathcal{W}(G)$ ; and any integer  $S' \geq 2\Lambda_M + 1$ , there exists an online algorithm (Algorithm 1) that returns a walk  $W \in \mathcal{W}(G)$  with expected reward*

$$\mathbb{E}[\text{Rew}(W, I, 1)] \geq \frac{1}{6\rho_M} \cdot \text{Rew}(W', I', S').$$

We begin by establishing some notation. Let  $W' \in \mathcal{W}(G)$  be any walk that services some predicted requests in  $I'$  with a service time of  $S'$ . We use  $\sigma'_i = (v'_i, r'_i, d'_i, \pi'_i)$  to denote the predicted requests in  $I'$  that are covered by  $W'$ , ordered in the sequence in which they are visited by  $W'$ . Likewise, we use  $\sigma_i = (v_i, r_i, d_i, \pi_i) \in I$  to denote the true request matched to the prediction  $\sigma'_i$ , that is,  $\sigma'_i = M(\sigma_i)$ .

At a high level, our algorithm follows the walk  $W'$ , but when it reaches a predicted request  $\sigma'_i$ , it considers taking a detour to service a true request that is available at that point of time. To this end, we define the set of “reachable” true requests as follows.

► **Definition 19.** *Given a partial walk  $W$  that is at request  $\sigma'_i \in I'$  at time  $t$ , we define the set of reachable requests  $R_i(W, t)$  to be the set of all  $\sigma \in I$  such that:*

1.  $r_\sigma \leq t \leq d_\sigma - \ell(v'_i, v_\sigma) - 1$ , and
2.  $2\ell(v'_i, v_\sigma) + 1 \leq S'$ .

Our algorithm considers all of the reachable requests that have not been covered by the walk as yet, chooses the one with the highest reward, and takes a detour to visit and cover the request, before returning to  $\sigma'_i$  and resuming the walk. In order to deal with time window errors, our algorithm starts the walk a little early, or on time, or a little late, as in the proof of Lemma 5. The algorithm is described below formally.

■ **Algorithm 1** Online algorithm for TSP-TW with predictions.

---

**Offline input:** Graph  $G$ , predicted requests  $I'$ , walk  $W' \in \mathcal{W}(G)$ , service times  $S'$ .

**Online input:** True requests  $I$ .

**Output:** Walk  $W \in \mathcal{W}(G)$ .

- 1: Let  $K = L_{\min}/2$ . Select  $\epsilon$  uniformly at random from  $\{-1, 0, 1\}$ .
  - 2: Define the set of covered requests  $C = \emptyset$ .
  - 3: **for**  $i \leftarrow 1$  to  $|\text{Cov}(W', I', S')|$  **do**
  - 4:   Let  $t'_i$  denote the time at which  $W'$  visits  $\sigma'_i$ .
  - 5:   Set  $t_i \leftarrow t'_i + \epsilon K$ .
  - 6:   Visit  $v'_i$  at time  $t_i$ .
  - 7:   Construct the set  $R_i(W, t_i)$  of requests in  $I$  reachable at time  $t_i$ .
  - 8:   **if**  $R_i(W, t_i) \setminus C = \emptyset$  **then**
  - 9:     Do nothing.
  - 10:   **else**
  - 11:     Let  $\hat{\sigma}$  be the highest reward request in  $R_i(W, t_i) \setminus C$ .
  - 12:     Visit  $v_{\hat{\sigma}}$ ; spend one unit of idle time at  $v_{\hat{\sigma}}$ ; return to  $v'_i$ .
  - 13:     Set  $C \leftarrow C \cup \{\hat{\sigma}\}$ .
- 

We begin our analysis by noting that the walk  $W$  constructed by the algorithm is always able to visit the vertices  $v'_i$  corresponding to requests  $\sigma'_i \in \text{Cov}(W', I', S')$  feasibly at the desired times  $t_i$ . This is because, by construction, the length of the detours that the walk  $W$  takes in Step 12 is always at most  $S'$  – the amount of idle time  $W'$  spends at  $v'_i$  – by virtue of the fact that  $\hat{\sigma} \in R_i(W, t_i)$  and therefore,  $2\ell(v'_i, v_{\hat{\sigma}}) + 1 \leq S'$ . Therefore, all of the requests  $\hat{\sigma}$  visited in Step 12 are indeed visited by the walk  $W$ .

We now relate the total reward covered by  $W$  to the reward contained in the true requests  $\sigma_i$  corresponding to  $\sigma'_i \in \text{Cov}(W', I', S')$ . To do so, we first note that with constant probability each such request is reachable by  $W$ .

► **Claim 20.** For each  $i$ ,  $\sigma_i \in R_i(W, t_i)$  with probability at least  $1/3$ .

Proof. Recall that by definition we have  $\sigma'_i = M(\sigma_i)$  and so,  $2\ell(v'_i, v_i) + 1 \leq 2\Lambda_M + 1 \leq S'$ . So the request  $\sigma$  always satisfies the second requirement in the definition of the reachable set  $R_i(W, t_i)$ . Let us now consider the first requirement and recall that  $t_i = t'_i + \epsilon K$  where  $\epsilon \in \{-1, 0, 1\}$ . We will now argue that  $t_i \in [r_i, d_i - 1 - \ell(v'_i, v_i)]$  for at least one of the three choices of  $\epsilon$ . The claim then follows from the uniformly random choice of  $\epsilon$ .

1. If  $t'_i \in [r_i, d_i - 1 - \ell(v'_i, v_i)]$ , then the claim holds for  $\epsilon = 0$  and  $t_i = t'_i$ .
2. Suppose that  $t'_i < r_i$ . Then, for  $\epsilon = 1$  we have that  $t_i = t'_i + K \geq r'_i + \tau_M \geq r_i$ , and also  $t_i = t'_i + K < r_i + K < d_i - L_{min} + L_{min}/2$  and thus  $t_i = t'_i + K \leq d_i - 1 - \ell(v'_i, v_i)$  since  $\ell(v'_i, v_i) \leq \Lambda_M \leq L_{min}/2$ . Thus, in this case we have  $t_i = t'_i + K \in [r_i, d_i - 1 - \ell(v'_i, v_i)]$  with the choice of  $\epsilon = 1$ .
3. Finally, suppose that  $t'_i > d_i - 1 - \ell(v'_i, v_i)$ . Then, for  $\epsilon = -1$  we have that  $t_i = t'_i - K \leq d'_i - S' - \tau_M \leq d_i - S'$  and thus  $t'_i - K \leq d_i - 1 - \ell(v'_i, v_i)$  since  $S' \geq 2\Lambda_M + 1 \geq \ell(v'_i, v_i) + 1$ . Also,  $t_i = t'_i - K > d_i - 1 - \ell(v_i, v'_i) - L_{min}/2 > r_i + L_{min}/2 - \ell(v_i, v'_i) - 1$  and thus  $t'_i - K \geq r_i$ , since  $\ell(v_i, v'_i) \leq \Lambda_M \leq L_{min}/2$ . Thus, in this case we have obtained that  $t_i = t'_i - K \in [r_i, d_i - 1 - \ell(v'_i, v_i)]$  with the choice of  $\epsilon = -1$ .  $\square$

We are now ready to prove Lemma 7 via a matching-type argument. To account for the reward covered by the walk  $W$  constructed by the algorithm, we will employ a standard charging scheme. Every time the algorithm takes a detour to cover some true request  $\hat{\sigma}$  from a predicted request  $\sigma'_i$  in Step 12, we will credit half of the earned reward  $\pi_{\hat{\sigma}}$  to  $\hat{\sigma}$  itself, and half of the reward to the request  $\sigma_i$ . Formally, let  $\text{Cr}(\sigma)$  denote the total credit received by  $\sigma \in I$ . Then during Step 12 we will increment both  $\text{Cr}(\hat{\sigma})$  and  $\text{Cr}(\sigma_i)$  by  $\pi_{\hat{\sigma}}/2$ .

Now consider some  $\sigma_i \in I$  corresponding to  $\sigma'_i \in \text{Cov}(W', I', S')$ . By Claim 20, this request is in  $R_i(W, t_i)$  with probability at least  $1/3$ . If at time  $t_i$ , the request has already been covered by  $W$ , then we get  $\text{Cr}(\sigma_i) \geq \pi_i/2$ . Otherwise, we pick a  $\hat{\sigma} \in R_i(W, t_i)$  with  $\pi_{\hat{\sigma}} \geq \pi_i$ , and therefore, once again we get  $\text{Cr}(\sigma_i) \geq \pi_i/2$ .

Putting everything together, we get

$$\begin{aligned} \mathbb{E}[\text{Rew}(W, I, S)] &= \mathbb{E}\left[\sum_{\sigma \in I} \text{Cr}(\sigma)\right] \geq \sum_{i: \sigma'_i \in \text{Cov}(W', I', S')} \mathbb{E}\left[\frac{\pi_i}{2} \mathbb{1}[\sigma_i \in R_i(W, t_i)]\right] \\ &\geq \frac{1}{3} \cdot \sum_{i: \sigma'_i \in \text{Cov}(W', I', S')} \frac{\pi_i}{2} \\ &\geq \frac{1}{6} \cdot \frac{1}{\rho_M} \cdot \sum_{i: \sigma'_i \in \text{Cov}(W', I', S')} \pi'_i \\ &= \frac{1}{6\rho_M} \cdot \text{Rew}(W', I', S') \end{aligned}$$

This completes the proof of the lemma.

## 7 Lower bounds

In this section, we present lower bounds that complement our results. First, we will motivate the need for predictions in Section 7.1. Then, in Section 7.2 we will show that the competitive ratio of TW-TSP with predictions must scale linearly with the error in locations. Finally, in Section 7.3 we argue the need for non-zero service times in the definition of TW-TSP with predictions.

### 7.1 Lower bounds for online TW-TSP without predictions

We argue that Online TW-TSP does not admit any reasonable competitive ratio in the absence of predictions. In the case of deterministic algorithms where their entire behavior is predictable, simple instances with only 2 vertices and appropriately small time-windows suffice to argue that no bounded guarantee for the approximation ratio is achievable.

► **Lemma 21.** *The competitive ratio of any deterministic online algorithm for Online TW-TSP on instances with  $L_{\min} \leq D$  is unbounded.*

**Proof.** Let DET be any deterministic algorithm and let  $G$  be the line graph with just two vertices  $v_1, v_2$  connected via an edge of length  $D$ . Since DET is deterministic, we can assume knowledge of its position at any step  $t$  as soon as we have specified all requests with release time  $\leq t$ . We will now construct a request sequence  $I$  that uses the information.

For the first  $D$  time-steps, we don't release any request. Then, at  $t = D$ , let  $v_D \in \{v_1, v_2\}$  be the position that DET's walk is currently at and likewise let  $v'_D$  be the other vertex of  $G$ . We construct the first request to be  $\sigma = (v'_D, D, D + L, 1)$  for any  $L \leq D$ . Clearly, DET cannot service this request as even for  $L = D$  it arrives on  $v'_D$  at deadline and cannot service it for one step. Then, we don't release any new request for the next  $2D$  steps, and at  $t = 3D$  we repeat the same process, by requesting the vertex that DET doesn't currently occupy. Likewise, we repeat the same process at  $t = 5D, t = 7D$  etc. Independently of the size of our request sequence, the total reward collected by DET is 0.

On the other hand, it is not hard to see that if our request sequence  $I$  has  $N$  requests in total, then  $\text{OPT}(G, I, 1) = N$ . This is due to the fact that requests are spaced  $2D$ -steps from each other, and thus an optimal offline algorithm that had knowledge of the entire sequence in advance would always be able to arrive at each request on time, servicing it within its respective time-window. ◀

For randomized algorithms, a slight improvement can be achieved. In particular, the randomized algorithm that picks a vertex uniformly at random and then remains idle on it for the entire sequence achieves a competitive ratio of  $1/n$ . It turns out that this is actually the best possible ratio that a randomized algorithm can achieve on Online TW-TSP:

► **Theorem 8.** *The competitive ratio of any randomized online algorithm for Online TW-TSP on instances with  $L_{\min} \leq D$  is at most  $1/n$ .*

**Proof.** Let  $G(V, E, \ell)$  be the uniform complete graph of  $n = |V|$  vertices, where all edges have a length of  $D$ . Fix any integer  $N$  and let vertices  $v_1, v_2, \dots, v_N \in V$  be drawn independently and uniformly at random. Next, we fix any window length  $L \leq D$  and consider the (random) request sequence on these vertices  $I = \{\sigma_i\}_{i=1}^N$  for  $\sigma_i = (v_i, (2i-1)D, (2i-1)D + L, 1)$ . From Yao's minimax principle, a lower bound on the (expected) competitive ratio of deterministic algorithms on this randomized instance will imply the same lower bound for randomized algorithms.

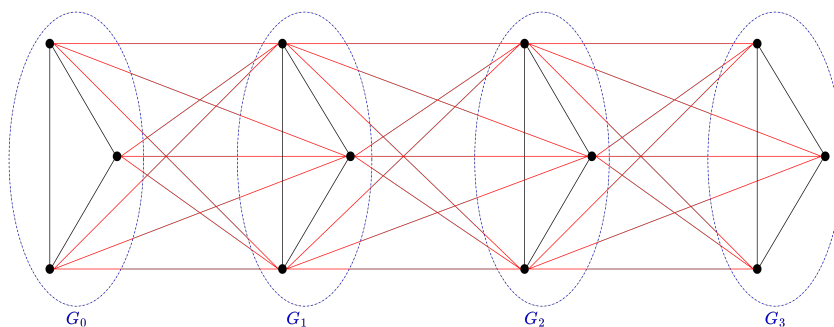
Since the time-windows are spaced  $2D$ -away from each other, it is not hard to see that for any realization of  $I$ ,  $\text{OPT}(G, I, 1) = N$ . On the other hand, since  $D \geq L$ , we get that the only way to service a request is to be on its corresponding vertex on release time. Since the vertices are random, for any deterministic algorithm this happens with probability precisely  $1/n$ , and thus the expected reward of any deterministic algorithm on this instance is  $N/n$ , proving the claim. ◀

## 7.2 Tight dependence on location error

In this section we show that a linear dependency on the location error is unavoidable for any randomized online algorithm for the TW-TSP with predictions, even assuming exponential computational power. In other words, we formally prove Theorem 10, which we re-state for the reader's convenience:

► **Theorem 10.** *For any  $S > 0$ , there exists an instance  $(G, I, I', M)$  of the TW-TSP with predictions satisfying  $\tau_M = 0$ ,  $\rho_M = 1$ , and  $\Lambda_M = S$  such that the competitive ratio of any randomized online algorithm taking the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input asymptotically approaches  $1/(S + 1)$ .*

**Proof.** Fix any  $S > 0$  and let  $K, C$  and  $N$  be integer parameters that will be specified later. We construct a graph  $G = \cup_{i=0}^{N-1} G_i$  that consists of  $N$  copies  $G_0, \dots, G_{N-1}$  of the complete graph on  $C$  vertices with all edge lengths equal to  $S$ , arranged in a way so that each vertex in  $G_i$  connects to each vertex in  $G_{i+1}$  with an edge of length  $KS$ . A pictorial example for small values of  $C$  and  $N$  is shown in Figure 1.



■ **Figure 1** An example of  $G$  for  $C = 3$  and  $N = 4$ . Black edges have a length of  $S$  and red edges have a length of  $KS$ .

Next, we select independently and uniformly at random one vertex  $v_i$  from each sub-graph  $G_i$  and construct the (randomized) request sequence  $I = \{\sigma_i\}_{i=0}^{N-1}$  where we denote  $\sigma_i = (v_i, r_i, r_i + KS, 1)$  for  $r_i = i \cdot (KS + 1)$ . As for the predictions, we simply construct a second instance  $I'$  in the same manner and offer it as an offline prediction for  $I$ . Observe that for all possible constructions of  $I$  and  $I'$  it holds that there exists a matching  $M$  between them with  $\tau_M = 0$ ,  $\rho_M = 1$  and  $\Lambda_M = S$ , namely the matching that pairs together vertices from the same sub-graphs. This prediction provides no information to the algorithm other than the fact that the (true) instance  $I$  was constructed via the above randomized approach. Also, notice that the location error  $\Lambda_M$  can be arbitrarily small compared to the window length  $L = KS$  by choosing appropriately large  $K$ .

It is easy to see that for all possible realizations of  $I$  it holds that  $\text{OPT}(G, I, 1) = N$ . Indeed, consider the walk that starts from the (random) vertex  $v_0$ , remains idle for 1 step and then visits vertex  $v_1$ , remains idle for one step, visits  $v_2$ , etc. Such a walk would visit each vertex  $v_i$  at step  $t = i \cdot KS + i = r_i$  and thus would service all the requests in  $I$ , achieving a total reward of  $N$ . Next, we will show that the expected reward of any deterministic algorithm on the random sequence  $I$  approaches  $N/S$ . Using Yao's minimax principle, this will immediately translate to a lower bound that approaches  $1/S$  for randomized algorithms, completing the proof of the theorem.

Fix any deterministic algorithm for TW-TSP with predictions on instance  $(G, I, I', M)$ . Note that since the predictions  $I'$  supply zero information, it suffices to analyze the algorithm as a deterministic online algorithm for Online TW-TSP on instance  $I$ . The key observation is that due to the fact that both the time windows of the requests and the edges that connect different sub-graphs have a length of  $KS$ , it is impossible for the algorithm to know on what vertex  $v_i$  of subgraph  $G_i$  the request is going to arrive before visiting some possibly different vertex of the same subgraph. In particular, if the algorithm is in subgraph  $G_j$  for  $j < i$  at time  $r_i$ , then it cannot reach vertex  $v_i$  before the time window of request  $i$  ends. Thus, in order for any deterministic algorithm to serve the request on sub-graph  $G_i$ , it first has to visit some vertex  $v'_i$  of  $G_i$ . If it so happens that  $v'_i = v_i$  then it can immediately service the request, otherwise it has to travel a distance of  $S$  in order to reach  $v_i$ .

We partition the set of requests into two sets  $N_+$  and  $N_-$  based on whether the deterministic algorithm happens to arrive on the correct vertex of the sub-graph or not. All the requests in  $N_+$  can be serviced without any extra delay, exactly as done by the optimal walk. On the other hand, servicing a request in  $N_-$  requires the algorithm to spend an extra time of  $S$  in order to transition to the right vertex. Since the time-windows have a length of  $KS$ , this can be done at most  $K$  times before the algorithm runs out of slack to spare. When this happens, the algorithm would have to skip the next  $S$  requests in order to recover enough slack to fix its next mistake.

Formally, consider the last request in  $N_-$  that the algorithm feasibly serves, call it  $l$ . Let  $A$  be the number of requests in  $N_-$  the algorithm serves through extra delay prior to  $l$ , and let  $B$  be the number of requests the algorithm skips in  $N_+$  or  $N_-$  prior to  $l$ . Then in order for the algorithm to have reached  $l$  before its time window ends, it must be the case that the total extra delay incurred by the algorithm, namely  $AS - B$ , is no more than  $KS$ . Rearranging we get:

$$A(S + 1) - (A + B) \leq KS, \quad \text{or,} \quad A \leq \frac{A + B}{S + 1} + \frac{KS}{S + 1} < \frac{N}{S + 1} + K$$

Thus, we get that the total expected reward gathered by the algorithm is at most

$$\mathbb{E}[|N_+|] + A + 1 \leq \mathbb{E}[|N_+|] + \frac{N}{S + 1} + K + 1$$

Finally, using the fact that  $\mathbb{E}[|N_+|] = N/C$ , we get that the competitive ratio of any deterministic algorithm on the random instance  $I$  is at most

$$\frac{1}{C} + \frac{K + 1}{N} + \frac{1}{S + 1}$$

Choosing  $N \gg K$  and  $C$  sufficiently large, we can make this competitive ratio asymptotically approach  $1/(S + 1)$  as desired.  $\blacktriangleleft$

### 7.3 Comparing the optimal with and without service times

As we saw in Lemma 4, the gap between  $\text{OPT}(G, I, 1)$  and  $\text{OPT}(G, I, S)$  depends linearly on the service time  $S$ . In this section, we study the gap between  $\text{OPT}(G, I, 1)$  and  $\text{OPT}(G, I, 0)$ , showing that there exists a much sharper separation between them.

► **Theorem 22.** *For any integers  $L$  and  $D$ ,  $L \leq D$ , there exists an offline instance  $(G, I)$  of the TW-TSP where  $D$  is the diameter of the network and every request has a time window length equal to  $L$ , such that  $\text{OPT}(G, I, 1) \leq \frac{L}{D+1} \cdot \text{OPT}(G, I, 0)$ .*

**Proof.** Consider the line graph  $G$  with vertices  $v_0$  through  $v_D$  connected sequentially via edges of length 1. Clearly, the diameter of  $G$  is  $D$ . Next, consider the sequence of  $(D + 1)$ -requests  $I = \{\sigma_i\}_{i=0}^D$  where  $\sigma_i = (v_i, i, L + i, 1)$ . Observe that  $\text{OPT}(G, I, 0) = D + 1$  as simply following the walk from  $v_0$  to  $v_D$  will cover all the requests if the service costs are 0.

On the other hand, it is not very hard to see that  $\text{OPT}(G, I, 1) = L$ . First, observe that without loss we can assume that the optimal walk starts on  $v_0$ . If not, let  $\sigma_i$  be the *first* request served by the optimal. Since  $\sigma_i$  cannot be served prior to step  $r_i = i$ , we could instead start at  $v_0$  and take  $i$  steps in order to reach  $v_i$  and then follow the original walk, achieving precisely the same reward.

Our argument is completed by the simple observation that any walk that starts from  $v_0$  and has served  $x$  requests with service cost 1 *cannot* visit vertex  $v_j$  prior to step  $j + x$ . Thus, as soon as  $x$  becomes  $L$  all the future time-windows will be missed. ◀

Theorem 22 states that  $\text{OPT}(G, I, 0)$  is a much stronger benchmark than  $\text{OPT}(G, I, 1)$ . However, it doesn't exclude the possibility of designing an algorithm that is competitive against this stronger benchmark  $\text{OPT}(G, I, 0)$ . We will next show that no randomized online algorithm with predictions can obtain a better than linear competitive ratio against this benchmark. Intuitively, requiring the algorithm to spend non-zero time at each request also allows the algorithm to recover from possible mistakes due to the prediction errors by skipping requests that the optimum services with a delay of 1. A similar approach is not possible in the 0 service time setting.

► **Theorem 9.** *For any  $S > 0$ , there exists an instance  $(G, I, I', M)$  of the TW-TSP with predictions and service times 0, satisfying  $\tau_M = 0$ ,  $\rho_M = 1$ , and  $\Lambda_M = S$ , such that any randomized online algorithm taking the tuple  $(G, I', \Lambda_M)$  as offline input and  $I$  as online input achieves a reward no larger than  $O(1/n) \cdot \text{OPT}(G, I, 0)$ . Here  $n$  is the number of vertices in  $G$ .*

**Proof.** We construct the same graph  $G = \cup_{i=0}^{N-1} G_i$  as in Theorem 10, that consists of  $N$  copies of the complete graph with edge lengths  $S$ , connected sequentially with edges of length  $KS$  (see Figure 1). As for the request sequence, we once again select independently and uniformly at random one vertex  $v_i$  from each sub-graph  $G_i$  and construct the (randomized) request sequence  $I = \{\sigma_i\}_{i=0}^{N-1}$  where  $\sigma_i = (v_i, iKS, (i + 1)KS - 1, 1)$ ; notice that release times are slightly different from Theorem 10 to account for the absence of service costs.

For the predictions, we simply construct a second instance  $I'$  in the same manner and offer it as an offline prediction for  $I$ . By matching the requests on the same sub-graph together, we get  $\tau_M = 0$ ,  $\rho_M = 1$  and  $\Lambda_M = S$ . As it clearly holds that  $\text{OPT}(G, I, 0) = N$  for any realization of  $I$ , to prove our theorem it suffices to argue that any deterministic algorithm for TW-TSP with predictions on instance  $(G, I, I', M)$  gets an expected reward of  $\mathcal{O}(N/n)$  and apply *Yao's minimax principle*. Furthermore, since the prediction  $I'$  does not provide any information on  $I$ , it suffices to bound the reward of deterministic algorithms for Online TW-TSP on (online) instance  $I$ .

Fix any deterministic algorithm for Online TW-TSP on instance  $I$ . Observe that since edges between different sub-graphs have a length of  $KS$  and time-windows have a length of  $KS - 1$ , it is impossible for the algorithm to service some request  $\sigma_i$  unless at  $t = r_i$  it is already in some vertex of sub-graph  $G_i$ . For the same reason, it is not possible to service any request  $\sigma_j$  after servicing some other request  $\sigma_i$  with  $i > j$ . Finally, since all requests can be reached by their release time if the algorithm starts from a vertex in  $G_0$ , we can assume without loss that this is indeed the case.

We partition our set of  $N$  requests into two sets  $N_+$  and  $N_-$  based on whether the deterministic algorithm *happens* to arrive on the correct vertex of the sub-graph before the request's release time or not. From the random construction of our instance, we have that  $\mathbb{E}[|N_+|] \leq N/C$ . For requests in  $N_-$ , if the algorithm wishes to service them it has to take a detour of length  $S$  in order to reach the correct vertex. Our proof relies on the fact that after servicing  $K$  requests in  $N_-$ , the algorithm can no longer service any other request. To see this, let  $v_j$  be the  $K$ -th request in  $N_-$  that was serviced by the deterministic algorithm. As we can already established, any request  $v_i$  with  $i < j$  can no longer be serviced. On the other hand, since without loss the algorithm starts at a vertex of  $G_0$ , just reaching a vertex in  $G_i$  while taking  $K$  detours of length  $S$  requires at least  $iKS + K > d_i$  time-steps. Putting everything together, we get that the expected reward of the algorithm is at most  $N/C + K = \mathcal{O}(N/n)$  by setting  $K = \mathcal{O}(1)$  and  $N = 2K$ , since  $n = NC$ . ◀

---

## References

- 1 Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- 2 Esther M. Arkin, Joseph B. M. Mitchell, and Giri Narasimhan. Resource-constrained geometric network optimization. In *Symposium on Computational Geometry (SoCG)*, 1998.
- 3 Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *Symposium on the Theory of Computing (STOC)*, 2021.
- 4 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021.
- 5 Yossi Azar and Adi Vardi. TSP with time windows and service time. *CoRR*, abs/1501.06158, 2015. [arXiv:1501.06158](https://arxiv.org/abs/1501.06158).
- 6 Étienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- 7 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Symposium on the Theory of Computing (STOC)*, 2004.
- 8 Reuven Bar-Yehuda, Guy Even, and Shimon Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *J. Algorithms*, 55:76–92, 2005.
- 9 Magnus Berg, Joan Boyar, Lene M. Favrholt, and Kim S. Larsen. Online minimum spanning trees with weight predictions. In *Workshop on Algorithms and Data Structures*, 2023.
- 10 Giulia Bernardini, Alexander Lindermayr, Alberto Marchetti-Spaccamela, Nicole Megow, Leen Stougie, and Michelle Sweering. A universal error measure for input predictions applied to online graph problems. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- 11 Avrim Blum, Shuchi Chawla, David Karger, Terran Lane, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. In *Foundations of Computer Science (FOCS)*, 2003.
- 12 Niv Buchbinder, Yaron Fairstein, Konstantina Mellou, Ishai Menache, and Joseph (Seffi) Naor. Online virtual machine allocation with lifetime and load predictions. In *International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2021.
- 13 Shuchi Chawla and Dimitris Christou. Online TSP with predictions. *CoRR*, abs/2304.01958, 2024. [arXiv:2304.01958](https://arxiv.org/abs/2304.01958).
- 14 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.



- 15 Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX)*, 2004.
- 16 Ke Chen and Sarel Har-Peled. The orienteering problem in the plane revisited. In *Symposium on Computational Geometry (SoCG)*, 2006.
- 17 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- 18 Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. Secretaries with advice. In *ACM Conference on Economics and Computation (EC)*, 2021. doi:10.1145/3465456.3467623.
- 19 Jon C. Ergun, Zhili Feng, Sandeep Silwal, David Woodruff, and Samson Zhou. Learning-augmented  $k$ -means clustering. In *International Conference on Learning Representations (ICLR)*, 2022.
- 20 Thomas Wilhelm Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schloter. Learning-augmented query policies for minimum spanning tree with uncertainty. In *Embedded Systems and Applications (ESA)*, 2022.
- 21 Jie Gao, Su Jia, Joseph S. B. Mitchell, and Lu Zhao. Approximation algorithms for time-window TSP and prize collecting TSP problems. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.
- 22 Themis Gouleakis, Konstantinos Lakis, and Golnoosh Shahkarami. Learning-augmented algorithms for online TSP on the line. *CoRR*, abs/2206.00655, 2022. arXiv:2206.00655.
- 23 Hsiao-Yu Hu, Hao-Ting Wei, Meng-Hsi Li, Kai-Min Chung, and Chung-Shou Liao. Online TSP with predictions. *CoRR*, abs/2206.15364, 2022. arXiv:2206.15364.
- 24 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.
- 25 Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2020.
- 26 Yoshiyuki Karuno and Hiroshi Nagamochi. 2-approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times. *Discret. Appl. Math.*, 129:433–447, 2003.
- 27 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.
- 28 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *European Symposium on Algorithms (ESA)*, 2021.
- 29 Alexander Lindermayr and Nicole Megow. Algorithms with predictions. URL: <https://algorithms-with-predictions.github.io/>.
- 30 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021.
- 31 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.
- 32 John N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22(3):263–282, 1992.
- 33 Chenyang Xu and Benjamin Moseley. Learning-augmented algorithms for online steiner tree. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 36:8744–8752, 2022.



# Degrees and Network Design: New Problems and Approximations

Michael Dinitz ✉

Johns Hopkins University, Baltimore, MD, USA

Guy Kortsarz ✉

Rutgers University, Camden, NJ, USA

Shi Li ✉

Nanjing University, Jiangsu, China

---

## Abstract

While much of network design focuses mostly on cost (number or weight of edges), node degrees have also played an important role. They have traditionally either appeared as an objective, to minimize the maximum degree (e.g., the Minimum Degree Spanning Tree problem), or as constraints that might be violated to give bicriteria approximations (e.g., the Minimum Cost Degree Bounded Spanning Tree problem). We extend the study of degrees in network design in two ways. First, we introduce and study a new variant of the Survivable Network Design Problem where in addition to the traditional objective of minimizing the cost of the chosen edges, we add a constraint that the  $\ell_p$ -norm of the node degree vector is bounded by an input parameter. This interpolates between the classical settings of maximum degree (the  $\ell_\infty$ -norm) and the number of edges (the  $\ell_1$ -degree), and has natural applications in distributed systems and VLSI design. We give a constant bicriteria approximation in both measures using convex programming. Second, we provide a polylogarithmic bicriteria approximation for the Degree Bounded Group Steiner problem on bounded treewidth graphs, solving an open problem from [17] and [12].

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis; Theory of computation → Routing and network design problems

**Keywords and phrases** Network Design, Degrees

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.3

**Category** APPROX

**Funding** *Michael Dinitz*: Supported in part by NSF awards CCF-1909111 and CCF-2228995.

## 1 Introduction

The overarching theme of network design problems is to find “inexpensive” subgraphs that satisfy some type of connectivity constraints. The notion of “inexpensive” is often either the number of edges (unweighted cost) or the sum of edge costs (weighted cost). However, it has long been recognized that in many applications *vertex degrees* matter as much (or more) than cost. This is particularly true in the context of networking and distributed systems, where the degree of a node often corresponds to the “load” on that node, as well as in VLSI design. So there has been a significant amount of work on handling degrees, either instead of or in addition to cost, which has led to many seminal papers and results. With degrees as an objective, these include the well known local search approach of Fürer and Raghavachari [8] for the Minimum Degree Spanning Tree problem and the Minimum Degree Steiner Tree problem. With degrees as a constraint, these include the iterative rounding [16] approach of Singh and Lau [23] for the Minimum-Cost Bounded-Degree Spanning Tree problem, as well as many extensions (most notably to Survivable Network Design with degree bounds [20], but see [19] for many other examples).



© Michael Dinitz, Guy Kortsarz, and Shi Li;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 3; pp. 3:1–3:17



Leibniz International Proceedings in Informatics

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

In this paper we extend the study of degrees in network design in two ways. First, we introduce what is (to the best of our knowledge) a new class of problems. Instead of bounding the cost and individual degrees as in [23, 20], our objective is to obtain minimum cost while satisfying a bound on the  $\ell_p$ -norm of the node degree vector. This interpolates between the maximum degree (the  $\ell_\infty$ -norm) and the total number of edges or unweighted cost (the  $\ell_1$ -degree). Second, we solve a well known open problem: We give a poly-logarithmic bicriteria approximation for the Group Steiner Tree problem with degree bounds on bounded treewidth graphs.

**$\ell_p$ -Objective.** While the maximum degree is often a reasonable objective, as is minimizing the total cost (either with or without degree bounds), there are many natural situations where none of these approaches are fully satisfactory. If we simply ignore the degrees and focus on cost (weighted or unweighted), then we might end up with a solution with highly imbalanced degrees, leading to large load at particular nodes. If we ignore costs and simply optimize the maximum degree, then we might return a solution with far more edges than are needed: if the structure of the graph forces some node to have large degree, then if we simply try to minimize the maximum degree we will not even try to make the degrees of other nodes small. Finally, optimizing under individual degree bounds implicitly assumes that nodes “really have” these degree bounds, i.e., they come from some external constraint. But this is of course not always the case: often we do not have real bounds on individual nodes, but rather a more vague desire to “keep degrees small”.

Hence we want some way of making sure that the maximum degree is small, but also encouraging few edges. A natural function that simultaneously accomplishes both of these goals is the  $\ell_p$ -norm of the degree vector, i.e., the function  $(\sum_{v \in V} (\deg_v)^p)^{1/p}$  for  $p \geq 1$  (and in particular for  $p = 2$ ), where  $\deg_v$  is the degree of  $v$  in the output subgraph. When  $p = 1$  this is simply (twice) the number of edges (i.e., the unweighted cost), and when  $p = \infty$  this is the maximum degree. But for intermediate values of  $p$ , it discourages very large degrees (in particular the maximum degree) since  $p > 1$  implies that large degrees have a larger effect on the norm than smaller degrees, while still also being effected on a non-trivial way by the smaller degrees. So we can either use the  $\ell_p$ -norm as an objective function, or we can use it as a constraint that is far more flexible than having simple degree constraints at every node.

This intuition, that the  $\ell_p$ -norm takes into account both the maximum and the distribution simultaneously, is one reason why the  $\ell_p$ -norm has been an important objective function for combinatorial problems. For example, the Set Cover problem was studied under the  $\ell_p$  norm of the vector of number of elements assigned to each set [10]. It was also extensively studied in scheduling problems (see for example [1, 2, 18, 15]). To the best of our knowledge, the  $\ell_p$  norm has not been studied in the context of network design, with the notable recent exception of *graph spanners* [6, 5], where the direct applications of spanners to distributed systems led to exactly this motivation. Similarly, MST with  $\ell_p$  norm is very important for VLSI design, since in many such settings we are forced to use spanning trees and hence the number of edges is fixed. So minimizing the  $\ell_p$  norm will likely derive a balanced degree vector which is of key importance for these VLSI application (see, for example, [24, 22]).

Motivated by the above discussion, we introduce and give the first approximation for the Survivable Network design problem with low cost under a bound on the  $\ell_p$  norm of the degree vector.

**Group Steiner Tree with Degree Bounds.** In addition to the study of  $\ell_p$ -norm problems, we also make significant progress on a known open problem: approximating Group Steiner Tree with degree bounds on bounded treewidth graphs. The Group Steiner Tree problem

(without degree bounds) is a classical optimization problem [9] which has played a central role in network design. In this problem there is a designated root node  $r$ , and a collection of (not necessarily disjoint) *groups* of vertices. The goal is to find a subtree which connects at least one vertex from each group to  $r$ , while minimizing the total cost of all edges in the subtree. The Degree Bounded Group Steiner problem was first raised by Hajiaghayi in [13] (in the 8th Workshop on Flexible Network Design), motivated by the online version of the problem and applications to VLSI design. In particular, while low cost is highly desirable, this cost is paid only once, while later the VLSI circuit is applied (evaluated) constantly. Low degrees imply that the computation of the value of the circuit can be done faster. See a discussion of why low degrees are important for Group Steiner in [17].

Unfortunately, despite significant recent interest in this problem [12, 17], progress has been elusive. In particular, polylogarithmic bicriteria approximations were not even known for simple classes such as *series-parallel* graphs, i.e., for graphs with treewidth 2. We go far beyond series-parallel graphs, and give results for bounded treewidth graphs.

## 1.1 Our Results and Techniques

We begin in Section 2 with a study of the  $\ell_p$ -Survivable Network Design problem. We are given the input graph  $G = (V, E)$ , with edge costs  $c \in \mathbb{R}_{\geq 0}^E$ . There is a connection requirement vector  $r \in \mathbb{Z}_{\geq 0}^{\binom{V}{2}}$ , a number  $p \geq 1$  and a bound  $A$  on the  $\ell_p$  norm of the degree vector of the output graph. The goal of the problem is to find the minimum-cost subgraph  $H$  of  $G$  satisfying the following:

- **(connection requirements)** for every  $u, v \in V$  with  $u \neq v$ , there are at least  $r_{u,v}$  edge disjoint paths between  $u$  and  $v$  in  $H$ , and
- **(degree constraint)**  $(\sum_{v \in V} d_H^p(v))^{1/p} \leq A$ , where  $d_H(v)$  is the degree of  $v$  in  $H$ .

We assume the input instance is feasible; that is, there is a valid sub-graph  $H$  satisfying both requirements. Let  $\text{opt}$  be the minimum cost of a valid subgraph  $H$ . The main theorem we prove for the problem is the following:

► **Theorem 1.** *There is a (randomized) algorithm which, given an instance of  $\ell_p$ -SURVIVABLE NETWORK DESIGN, outputs a subgraph  $H$  satisfying the connection requirements and which has the following properties.*

- *The expected cost of  $H$  is at most  $2 \cdot \text{opt}$ .*
- *The expectation of the  $\ell_p$ -norm of the degree vector is at most  $2^{1/p} 5^{1-1/p} \cdot A$ .*

For the special case of  $\ell_p$ -Spanning Tree problem, where  $r_{uv} = 1$  for all  $u, v \in V$ , we improve the expected cost to at most  $\text{opt}$  (rather than  $2 \cdot \text{opt}$ ) and the expectation of the  $\ell_p$ -norm of the degree vector to at most  $2^{1-1/p} \cdot A$ .

Our main approach is to leverage the fact that the  $\ell_p$ -norm is *convex*. This allows us to write a convex relaxation for the problem, which can then be solved efficiently using standard convex programming techniques. We then round this solution using an iterative rounding approach. Making this work requires overcoming a number of issues, possibly the trickiest of which is handling fractional degrees that are less than 1. Note that a fractional solution could have many nodes with very small fractional degree (e.g.,  $1/n$ ). Due to the structure of the  $\ell_p$ -norm, such small values contribute far less to the  $\ell_p$ -norm than they “should” (in an integral solution). To get around this, we actually *change* the  $\ell_p$ -constraint in a way that acts differently for values less than 1, while still maintaining convexity. With this change in place, we can solve the relaxation, interpret the fractional degrees “as if” they are true degree bounds, and then round using existing results on iterative rounding for degree-bounded network design.

We then move to our second problem, Group Steiner Tree with Degree Bounds on bounded treewidth graphs. In the problem, we are given a graph  $G = (V, E)$  with treewidth  $\text{tw}$ , a cost vector  $c \in \mathbb{R}_{\geq 0}^E$ , a root  $r$ , and  $k$  sets  $S_1, S_2, \dots, S_k$ . We are additionally given a degree bound  $\text{db}_v \in \mathbb{Z}_{>0}$  for every  $v \in V$ . The goal of the problem is to choose a minimum-cost subgraph  $H$  of  $G$  such that for every  $t \in k$ ,  $H$  contains a path from  $r$  to some vertex in  $S_t$ , and  $d_H(v) \leq \text{db}_v$  for every  $v \in V$ . By minimality, the optimum  $H$  is always a tree. We solve an open problem from [12] and [17] by giving a polylogarithmic bicriteria algorithm as long as the treewidth is bounded. In particular, we prove the following theorem.

► **Theorem 2.** *There is an  $n^{O(\text{tw} \log \text{tw})}$ -time randomized algorithm for the Group Steiner Tree with Degree Bounds problem on bounded treewidth graphs which has  $O(\log^2 n)$  approximation ratio and  $O(\log^2 n)$ -degree violation.*

In order to achieve this result, we reduce the problem to a “tree labeling” problem in Section 3, which has been used in prior results [11, 12, 21]. There is a rooted full binary tree, and we need to give a label  $\ell_u$  for each node  $u$  in the tree from a subset  $L_u$  of potential labels. For every internal node  $u$  with two children  $v$  and  $v'$  there are some consistency constraints on the labels, which say that the triple  $(\ell_u, \ell_v, \ell_{v'})$  must be from some given subset  $\Gamma_u \in L_u \times L_v \times L_{v'}$ . Then we have some covering constraints, each specified by a set  $S$  of labels: the constraint requires that at least one node has its label in  $S$ . Finally, we have many cost constraints. For each such constraint, a label is given a cost, and we require that the total cost of all labels used is at most 1. For this problem we give a randomized algorithm that outputs a labeling that satisfies all consistency constraints, and approximately satisfies the covering and cost constraints with reasonable probability, assuming the given instance is feasible. It runs in polynomial time when the depth of the tree is  $O(\log n)$  and each  $L_u$  has  $O(1)$ -size. The main techniques of the algorithm are adaptations of the LP-rounding algorithm in [12] for their degree-bounded network design problem. We introduce the tree labeling problem as a host for these techniques, and adapt them for the problem.

Due to space limitations, we leave the reduction of Group Steiner Tree with Degree Bounds on bounded treewidth graphs to this tree labeling problem to Appendix B, and give a high-level idea here. Let  $\text{tw}$  be the treewidth of the graph; it is known from [3] that we can assume the decomposition tree of  $G$  is an  $O(\log n)$ -depth binary tree, with bag size  $O(\text{tw})$ . This decomposition tree will be the tree in the tree-labeling instance. For each bag in the tree, a label will contain the set of edges we take from the bag, and some connectivity information on the vertices in the bag. We define the consistency constraints so that if they are satisfied, then the connectivity information is correct. A group being connected can be captured by a covering constraint in the tree labeling instance, and the edge cost constraint and degree constraints can be formulated as cost constraints in the instance. Using the algorithm for the tree labeling instance, we obtain a tree with small cost that satisfies degree bounds approximately, and connects a group with reasonable probability. The final output then is obtained by running the procedure many times and taking the union.

## 1.2 Other Related Work

For the survivable network design problem without any degree constraints, the classic result of Jain [16] gives a 2-approximation algorithm using the iterative rounding method. In [9] an  $O(\log^2 n)$  approximation is given for the Group Steiner problem on tree inputs, and an  $O(\log^3 n)$  for the Group Steiner problem (without degree constraints) for bounded treewidth graphs. The approximation for trees is almost the best possible, unless NP problems can be solved in quasi-polynomial time [14]. [12] gave a bicriteria approximation for the Group

Steiner Tree Problem with degree bounds on tree inputs, with approximation ratio  $O(\log^2 n)$  and degree violation  $O(\log n)$ . Both bounds are nearly optimal [14, 7]. In [4] the authors gave an  $O(\log^2 n)$ -approximation ratio for Group Steiner problem on bounded treewidth graphs (without degree bounds). In [17] an  $O(\log^2 n)$  approximation is given for the Group Steiner problem with minimum maximal degree, but without costs.

**Notations.** Given a graph  $H$  and a vertex  $v$  in  $H$ , we shall use  $\delta_H(v)$  to denote the set of edges in  $H$  incident to  $v$ , and  $d_H(v) = |\delta_H(v)|$  to denote its degree. Given a rooted tree  $T$  and a vertex  $v$  in  $T$ , we use  $\Lambda_T(v)$  to denote the set of children of  $v$  in  $T$ , and  $\Lambda_T^*(v)$  to denote the set of descendants of  $v$  in  $T$  (including  $v$  itself). When  $H$  and  $T$  are clear from the context, we shall omit them in the subscript. For example, this happens when  $H = G$  is the input graph. For a real vector  $z$  over some domain, and a subset  $S$  of elements in the domain, we define  $z(S) := \sum_{i \in S} z_i$  to denote the sum of  $z$  values of elements in  $S$ .

## 2 $\ell_p$ -Survivable Network Design

In this section, we give our iterative rounding algorithm for  $\ell_p$ -survivable network design problem. Recall that we are given a graph  $G = (V, E)$  with cost vector  $c \in \mathbb{R}_{\geq 0}^E$ , a connection requirement vector  $r \in \mathbb{Z}_{\geq 0}^{\binom{V}{2}}$ , and a bound  $A$  on the  $\ell_p$  norm of the degree vector.

► **Definition 3.** We say a polytope  $\mathcal{P} \in [0, 1]^E$  is good if it is upward-closed<sup>1</sup> and the following holds: For every vector  $x \in \{0, 1\}^E$ , we have that  $x \in \mathcal{P}$  if and only if the graph  $(V, \{e \in E : x_e = 1\})$  satisfies the connection requirements.

Notice that the above definition does not capture the degree constraints. This is done using the following definition. For a real vector  $B \in [1, \infty]^V$ , we define  $\mathcal{Q}_B := \{x \in [0, 1]^E : \forall v \in V, x(\delta(v)) \leq B_v\}$  to be the set of all vectors satisfying the degree bounds defined by  $B$ .

► **Definition 4.** Let  $\alpha \geq 1$  and  $\beta \geq 0$  be two real numbers and  $\mathcal{P}$  be a good polytope. We say  $\mathcal{P}$  is  $(\alpha, \beta)$ -integral if for every  $B \in [1, \infty]^V$ , every non-integral extreme point  $x$  of  $\mathcal{P} \cap \mathcal{Q}_B$  satisfies at least one of the following two properties:

(4a) there exists an edge  $e \in E$  with  $1/\alpha \leq x_e < 1$ ,

(4b) there is a vertex  $v \in V$  such that  $x(\delta(v)) = B_v$  and  $|\{e \in \delta(v) : x_e > 0\}| \leq B_v + \beta$ .

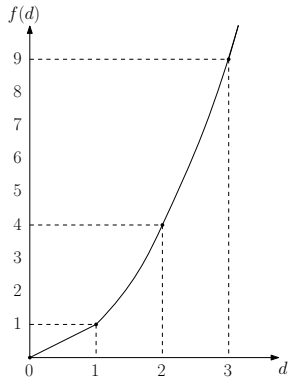
It is well known that for Survivable Network Design there is a  $(2, 3)$ -integral polytope  $\mathcal{P}$  [20]. For the special case of spanning tree, i.e.,  $r \equiv 1$ , there is a  $(1, 1)$ -integral polytope [23].

We will use these polytopes in our algorithm, and will show that their existence implies good approximation algorithms. More formally, we prove the following theorem.

► **Theorem 5.** Assuming the existence of an  $(\alpha, \beta)$ -integral polytope, there is a randomized algorithm which outputs a subgraph  $H$  of  $G$  satisfying the connection requirements. The expected cost of  $H$  is at most  $\alpha \cdot \text{opt}$  and the expectation of the  $p$ -norm of degree vector is at most  $\alpha^{1/p}(\alpha + \beta)^{1-1/p} A$ ; recall that  $\text{opt}$  is the value of the instance.

Note that this theorem, together with the existence of a  $(2, 3)$ -integral polytope for the general case and a  $(1, 1)$ -integral polytope for the spanning tree case, imply Theorem 1. So we focus on proving Theorem 5.

<sup>1</sup> This means for every  $x \in \mathcal{P}$  and  $x' \in [0, 1]^E$  with  $x' \geq x$ , we have  $x' \in \mathcal{P}$ .

(a) The function  $f$  for  $p = 2$ .

- 1: Solve LP(1) to obtain a solution  $x$
- 2: let  $B_v \leftarrow \max\{x(\delta(v)), 1\}$  for every  $v \in V$
- 3: **while true do**
- 4:   randomly choose an extreme point  $x'$  of  $\mathcal{P} \cap \mathcal{Q}_B$  such that  $\mathbb{E}[x'] = x$
- 5:    $x \leftarrow x'$
- 6:   **if**  $x$  is integral **then return**  $x$
- 7:   **if** case a happens for some  $e = (u, v) \in E$  **then**
- 8:        $x_e \leftarrow 1, B_v \leftarrow x(\delta(v)), B_u \leftarrow x(\delta(u))$
- 9:   **else**                            $\triangleright$  case b happens for some  $v$
- 10:      $B_v \leftarrow \infty$

(b) Iterative Rounding Algorithm for Network Design.

■ **Figure 1** The function  $f$  and the iterative rounding algorithm.

## 2.1 The Convex Program

Define a function  $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  as follows:  $f(x) = \begin{cases} x & \text{if } x \in [0, 1] \\ x^p & \text{if } x > 1 \end{cases}$ . Figure (1a) shows this function for  $p = 2$ . This is a convex function for  $p \geq 1$ .

Let  $\mathcal{P}$  be an  $(\alpha, \beta)$ -integral polytope. The following is our convex programming relaxation for the problem:

$$\min \sum_{e \in E} c_e x_e \quad \text{s.t.} \quad x \in \mathcal{P}, \quad \sum_{v \in V} f(x(\delta(v))) \leq A^p. \quad (1)$$

Recall that using our notation,  $x(\delta(v))$  is the sum of  $x$  values of edges incident to  $v$  in  $G$ . (1) is a convex program and can be solved efficiently. Since the indicator vector of the optimum subgraph  $H$  satisfies all the constraints, the value of the convex program is at most opt.

We note that if we instead used the function  $f(x) = x^p$  (i.e., without handling the  $0 \leq x \leq 1$  case separately), we would still have a convex relaxation of our problem. However, it is not hard to show that this relaxation has an extremely large integrality gap (even if we are allowed to violate the  $\ell_p$ -norm constraint by a polylogarithmic factor). Treating  $0 \leq x \leq 1$  differently from  $x > 1$  is one of the key ideas in our approximation algorithm.

## 2.2 The Iterative Rounding Algorithm

Our iterative rounding algorithm is described in Figure (1b). In Step 1, we solve the convex relaxation (1) to obtain an extreme solution  $x$ , which can be done in polynomial time using standard techniques. Then in Step 2 we define  $B_v = \max\{x(\delta(v)), 1\}$  for every  $v$  to be the upper bound on the degree of  $v$ . So, before Loop 3, we have  $x \in \mathcal{P} \cap \mathcal{Q}_B$ . We shall maintain this property before and after each iteration of the loop.

In each iteration of Loop 3, we randomly choose a vertex point  $x'$  of  $\mathcal{P} \cap \mathcal{Q}_B$  such that  $\mathbb{E}[x'] = x$  (Step 4) and then update  $x$  to be the  $x'$  (Step 5). This is possible since at the beginning of the iteration we have  $x \in \mathcal{P} \cap \mathcal{Q}_B$ . If  $x$  is integral, we then return  $x$  in Step 6. If we did not return, by that  $\mathcal{P}$  is  $(\alpha, \beta)$ -integral, either a or b happens. In the former case, we update  $x_e$  to 1, and change  $B_v$  and  $B_u$  for the two end vertices  $u, v$  of  $e$  so that we still have  $B_{v'} = \max\{x(\delta(v')), 1\}$  for every  $v' \in V$  (Step 8). In the latter case, we change  $B_v$  to  $\infty$  so that there will be no degree constraint for  $v$  from now on. Notice that in either case, we maintain the invariant that  $x \in \mathcal{P} \cap \mathcal{Q}_B$  as  $\mathcal{P}$  is upward-closed.



Notice that once  $x_e$  becomes 0 or 1 in some iteration, it will remain unchanged. This holds since for  $\mathbb{E}[x'_e] = x_e \in \{0, 1\}$  to hold, we must always have  $x'_e = x_e$ . When the algorithm terminates, it returns an integral  $x$  which satisfies the connectivity requirements. This holds since we have  $x \in \mathcal{P}$  and  $\mathcal{P}$  is good. The algorithm will terminate in  $O(|E|)$  iterations since in every iteration, we either fixed the value of some  $x_e$  to 1, or changed some  $B_v$  from a finite number to  $\infty$ .

### 2.3 Analysis of the Algorithm

We now begin to analyze the algorithm. As discussed, the algorithm will terminate with a subgraph which satisfies the connectivity requirements. To prove Theorem 5, we need to analyze the total cost and the  $\ell_p$ -norm of the degrees.

In Step 8, we say that we *round* the edge  $e$ . In Step 10, we say we *relax* the vertex  $v$ . At any time of the algorithm, we define a vector  $\bar{x} \in [0, 1]^E$  as follows. If  $e$  has not been rounded yet, then let  $\bar{x}_e = x_e$ . Otherwise, let  $\bar{x}_e$  be the value of  $x_e$  right before Step 8 in which we round  $e$ . Thus, from the moment,  $\bar{x}_e$  remains unchanged.

Let  $T$  be the number of iterations we run Loop 3; notice that this is a random variable. For every integer  $t \in [0, T]$ , we let  $x^t, \bar{x}^t, B^t$  to be the values of  $x, \bar{x}, B$  at the end of the  $t$ -th iteration of the Loop 3. So  $x^T$  is the output of the algorithm.

► **Observation 6.** *The following statements are true.*

- (6a) *During Loop 3, we always have  $\bar{x}_e \leq x_e \leq \alpha \bar{x}_e$  for every  $e \in E$ .*
- (6b) *Assume  $x^0(\delta(v)) < 1$  for a vertex  $v \in V$ . Then at the first moment when  $x(\delta(v)) \geq 1$  holds, we have  $\bar{x}(\delta(v)) \leq 1$ .*
- (6c)  *$\bar{x}(\delta(v))$  does not change from the first moment  $x(\delta(v)) \geq 1$  holds, until the moment  $v$  is relaxed, or the end of the algorithm if this does not happen.*

**Proof.**  $\bar{x}_e \leq x_e$  by the definition of  $\bar{x}_e$ . Moreover,  $x_e \leq \alpha \bar{x}_e$  as if  $x_e > \bar{x}_e$ , then  $x_e = 1$  and  $x_e \geq \frac{1}{\alpha}$ . So a holds.

To prove b, we consider two scenarios. In the first scenario, the moment is after Step 5 in some iteration. In this scenario,  $\bar{x}(\delta(v)) = x(\delta(v)) = 1$  since  $B_v = 1$  at the moment. In the second scenario, the moment is after we round some edge  $e \in \delta(v)$  in Step 8. In this case  $\bar{x}(\delta(v))$  is the same as  $x(\delta(v))$  before the step, which is strictly less than 1.

c holds since we maintained  $B_v = x(\delta(v))$  from the moment  $x(\delta(v))$  becomes at least 1. If  $\bar{x}_e \neq x_e$  at some time, it must be the case that  $x_e = 1$ . In this case, both  $x_e$  and  $\bar{x}_e$  will not change in the future. ◀

We can now analyze the expected cost of the algorithm. First, though, we will need a structural result.

► **Lemma 7.** *For every edge  $e \in E$ , the sequence  $\bar{x}_e^0, \bar{x}_e^1, \dots, \bar{x}_e^T$  is a martingale.*

**Proof.** Focus on an iteration  $t \geq 1$  and edge  $e \in E$ , and we fix the sequence  $\bar{x}_e^0, \bar{x}_e^1, \dots, \bar{x}_e^{t-1}$ . For simplicity we use  $\mathbb{E}'[\cdot]$  to denote  $\mathbb{E}[\cdot | \bar{x}_e^0, \bar{x}_e^1, \dots, \bar{x}_e^{t-1}]$ . We need to prove  $\mathbb{E}'[\bar{x}_e^t] = \bar{x}_e^{t-1}$ .

If we rounded  $e$  in iteration  $t$  or before, then  $\bar{x}_e^t = \bar{x}_e^{t-1}$  happens with probability 1. So, we can assume that  $e$  has not been rounded by the end of iteration  $t$ . In this case,  $\bar{x}_e^{t-1} = x_e^{t-1}$ .

So, in iteration  $t$ , either a happens for some  $e' \neq e$ , or b happens. In either case, we have  $\mathbb{E}'[\bar{x}_e^t] = \mathbb{E}'[x_e^t] = x_e^{t-1} = \bar{x}_e^{t-1}$  by the way we define the distribution for  $x'$  in Step 4. Therefore,  $\bar{x}_e^0, \bar{x}_e^1, \dots, \bar{x}_e^T$  is a martingale. ◀

► **Corollary 8.**  $\mathbb{E} \left[ \sum_{e \in E} c_e x_e^T \right] \leq \alpha \sum_{e \in E} c_e x_e^0$ .

**Proof.**

$$\mathbb{E} \left[ \sum_{e \in E} c_e x_e^T \right] \leq \alpha \mathbb{E} \left[ \sum_{e \in E} c_e \bar{x}_e^T \right] = \alpha \sum_{e \in E} c_e \bar{x}_e^0 = \alpha \sum_{e \in E} c_e x_e^0.$$

The inequality is by a and the first equality used Lemma 7.  $\blacktriangleleft$

Now that we understand the expected cost, it only remains to analyze the degree constraint. From now on we fix a vertex  $v \in V$ . We upper bound  $x^T(\delta(v))$ , which will in turn give an upper bound on  $\mathbb{E} [(x^T(\delta(v)))^p]$ . The main lemma we prove is

**► Lemma 9.** *For every  $v \in V$ , we have  $\mathbb{E} [(x^T(\delta(v)))^p] \leq \alpha(\alpha + \beta)^{p-1} \cdot f(x^0(\delta(v)))$ .*

**Proof.** We first consider the case  $x^0(\delta(v)) \geq 1$ . Let  $t$  be the iteration in which  $v$  is relaxed, or let  $t = T$  if  $v$  is not relaxed during the algorithm. By Property c,  $\bar{x}(\delta(v))$  does not change until the end of iteration  $t$ . Then, we have  $x^T(\delta(v)) \leq x^t(\delta(v)) + \beta \leq \alpha \bar{x}^t(\delta(v)) + \beta = \alpha \bar{x}^0(\delta(v)) + \beta = \alpha x^0(\delta(v)) + \beta$ . Notice that this happens with probability 1.

Notice that  $\mathbb{E}[x^T(\delta(v))] \leq \alpha \mathbb{E}[\bar{x}^T(\delta(v))] = \alpha \bar{x}^0(\delta(v)) = \alpha x^0(\delta(v))$  by Lemma 7. We have:

$$\begin{aligned} \mathbb{E} [(x^T(\delta(v)))^p] &\leq \frac{\alpha x^0(\delta(v))}{\alpha x^0(\delta(v)) + \beta} (\alpha x^0(\delta(v)) + \beta)^p = \alpha x^0(\delta(v)) (\alpha x^0(\delta(v)) + \beta)^{p-1}, \\ \frac{\mathbb{E} [(x^T(\delta(v)))^p]}{f(x^0(\delta(v)))} &\leq \frac{\alpha x^0(\delta(v)) (\alpha x^0(\delta(v)) + \beta)^{p-1}}{(x^0(\delta(v)))^p} = \alpha \left( \alpha + \frac{\beta}{x^0(\delta(v))} \right)^{p-1} \leq \alpha(\alpha + \beta)^{p-1}. \end{aligned}$$

Now we consider the second case:  $x^0(\delta(v)) < 1$ . We prove that with probability 1, we have  $x^T(\delta(v)) \leq \alpha + \beta$ . Assume  $x(\delta(v)) \geq 1$  happens at some time of the algorithm. By b, at the first moment when  $x(\delta(v)) \geq 1$ , we have  $\bar{x}(\delta(v)) \leq 1$ . By c, from the moment until the moment  $v$  becomes relaxed (or until the end of the algorithm if  $v$  is never relaxed),  $\bar{x}(\delta(v))$  does not change. Therefore, immediately after  $v$  becomes relaxed, we have  $\bar{x}(\delta(v)) \leq 1$ . Thus  $x^T(\delta(v))$  is at most the value of  $x(\delta(v)) + \beta$  at this moment, which is at most  $\alpha \bar{x}(\delta(v)) + \beta \leq \alpha + \beta$ . If  $x(\delta(v)) \geq 1$  never happens, then  $x^T(\delta(v)) < 1 \leq \alpha + \beta$ .

As in the first case, we have  $\mathbb{E}[x^T(\delta(v))] \leq \alpha x^0(\delta(v))$ . So

$$\begin{aligned} \mathbb{E} [(x^T(\delta(v)))^p] &\leq \frac{\alpha x^0(\delta(v))}{\alpha + \beta} (\alpha + \beta)^p = \alpha x^0(\delta(v)) (\alpha + \beta)^{p-1}, \\ \frac{\mathbb{E} [(x^T(\delta(v)))^p]}{f(x^0(\delta(v)))} &\leq \frac{\alpha x^0(\delta(v)) (\alpha + \beta)^{p-1}}{x^0(\delta(v))} = \alpha(\alpha + \beta)^{p-1}. \end{aligned}$$

So, we always have  $\mathbb{E} [(x^T(\delta(v)))^p] \leq \alpha(\alpha + \beta)^{p-1} f(x^0(\delta(v)))$ . This implies  $\mathbb{E} [\sum_v (x^T(\delta(v)))^p] \leq \alpha(\alpha + \beta)^{p-1} A^p$ .  $\blacktriangleleft$

Corollary 8 and Lemma 9 imply Theorem 5, which in turn implies Theorem 1.

### 3 A Tree Labeling Problem

In this section, we describe the tree labeling problem to which we reduce the Group Steiner Tree problem with degree bounds on bounded-treewidth graphs; the problem has served as a building block in many prior results [11, 12, 21]. We are given a full binary tree  $\mathbf{T} = (\mathbf{V}, \mathbf{E})$  rooted at  $\mathbf{r} \in \mathbf{V}$ .<sup>2</sup> For every vertex  $u \in \mathbf{V}$ , we are given a finite set  $L_u$  of labels for  $u$ ; we

<sup>2</sup> It is not important to require the binary tree to be full; our algorithm works when some internal nodes have only one child. Assuming every internal node have 2 children is only for notational convenience.

assume  $L_u$ 's are disjoint and let  $L := \bigcup_{u \in \mathbf{V}} L_u$ . The output is a labeling  $\vec{\ell} = (\ell_u \in L_u)_{u \in \mathbf{V}}$  of the vertices  $\mathbf{V}$ , that satisfies the constraints described below.

- **(consistency constraints)** For every internal node  $u$  of  $\mathbf{T}$  with two children  $v$  and  $v'$ , we are given a set  $\Gamma_u \subseteq L_u \times L_v \times L_{v'}$ . A valid labeling  $\vec{\ell}$  must satisfy  $(\ell_u, \ell_v, \ell_{v'}) \in \Gamma_u$ .
- **(covering constraints)** We are given  $k$  subsets  $S_1, S_2, \dots, S_k \subseteq L$ . A valid labeling  $\vec{\ell}$  needs to satisfy that for every  $t \in [k]$ ,  $\ell(\mathbf{V}) \cap S_t \neq \emptyset$ , where  $\ell(\mathbf{V})$  is defined as  $\{\ell_u : u \in \mathbf{V}\}$ . In words,  $\ell(\mathbf{V})$  needs to intersect every  $S_t$ .
- **(cost constraints)** We are given  $m \geq 0$  linear constraints defined by the costs  $(c_\ell^i \in [0, 1])_{i \in [m], \ell \in L}$ . For every  $i \in [m]$ , a valid labeling  $\vec{\ell}$  needs to satisfy  $\sum_{u \in \mathbf{V}} c_{\ell_u}^i \leq 1$ . In words, there are  $m$  types of resource, and we have 1 unit of each type. Setting the label of  $u$  to  $\ell$  will use  $c_\ell^i$  units of type  $i$ -resource.

We say a labeling  $\vec{\ell} = (\ell_u \in L_u)_{u \in \mathbf{V}}$  is *consistent* if it satisfies the consistency constraints. Given a consistent labeling  $\vec{\ell}$ , we say it covers group  $S_t$  if  $\ell(\mathbf{V}) \cap S_t \neq \emptyset$ . We define its type- $i$  cost to be  $\text{cost}^i(\vec{\ell}) := \sum_{u \in \mathbf{V}} c_{\ell_u}^i$ . So a valid labeling  $\vec{\ell}$  for the instance is a consistent one that covers all groups, and has  $\text{cost}^i(\vec{\ell}) \leq 1$  for every  $i \in [m]$ .

Given a label tree instance, we let  $n = |\mathbf{V}|$ ,  $D$  be the height of  $\mathbf{T}$  (the maximum number of edges in a root-to-leaf path in  $\mathbf{T}$ ) and  $\Delta = \max_{u \in \mathbf{V}} |L_u|$  be the maximum size of any  $L_u$ . The main theorem we prove is the following:

► **Theorem 10.** *Assume we are given a feasible label tree instance  $(\mathbf{T} = (\mathbf{V}, \mathbf{E}), \mathbf{r}, (L_u)_u, (\Gamma_u)_u, (S_t)_{t \in [k]}, A \in [0, 1]^{m \times L})$ , i.e., there is a valid labeling. There is a randomized algorithm that in time  $\text{poly}(n) \cdot \Delta^{O(D)}$  outputs a consistent labeling  $\vec{\ell}$  such that the following holds.*

**(10a)** *For every  $t \in [k]$ , we have  $\Pr[\vec{\ell} \text{ covers group } S_t] \geq \frac{1}{D}$ .*

**(10b)** *For every  $i \in [m]$ , we have  $\mathbb{E} \left[ \exp \left( \ln \left( 1 + \frac{1}{2D} \right) \cdot \text{cost}^i(\vec{\ell}) \right) \right] \leq 1 + \frac{1}{D}$ .*

Property b gives a tail concentration bound on  $\text{cost}^i(\vec{\ell})$ . The remaining part of this section is dedicated to the proof of Theorem 10.

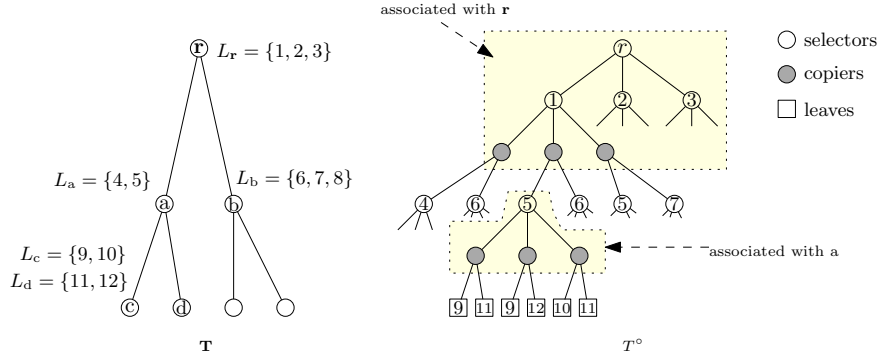
### 3.1 Construction of a super-tree $T^\circ$

In this section, we construct a rooted tree  $T^\circ = (V^\circ, E^\circ)$  of size  $O(n)\Delta^{O(D)}$  such that a consistent labeling of  $\mathbf{T}$  corresponds to what we call a *consistent sub-tree*. So we can reduce the problem to finding the latter object. The root of  $T^\circ$  is  $r$ . Each internal node of  $T^\circ$  is either a *selector node*, or a *copier node*; their meanings will be clear soon. Each node  $p \in V^\circ$  is *associated with* a node  $u$  in  $\mathbf{T}$ . Each non-root selector node or leaf node is *associated with* a label  $\ell \in L_u$ . We shall use  $p$  and  $q$  and their variants to denote nodes in  $T^\circ$ , and  $u$  and  $v$  and their variants to denote nodes in  $\mathbf{T}$ .

The algorithm for constructing  $T^\circ$  is described in Algorithm 1, which calls the procedure `construct-tree` described in Algorithm 2. See Figure 2 for the illustration of the construction of  $T^\circ$  from  $\mathbf{T}$ . For a node  $p \in V^\circ$ , we use  $\Lambda(p)$  denotes the set of children of  $p$  in  $T^\circ$ , and  $\Lambda^*(p)$  denotes the set of descendants of  $p$  in  $T^\circ$ , including  $p$  itself.

■ **Algorithm 1** Main algorithm for the construction of  $T^\circ$ .

- 
- 1: create a node  $r$  associated with  $\mathbf{r}$  as the root of  $T^\circ$ , and let  $r$  be a *selector node*
  - 2: **for** every  $\ell \in L_{\mathbf{r}}$  **do**:
  - 3:     create a child  $p$  of  $r$ , associated with node  $\mathbf{r}$  and label  $\ell$
  - 4:     call `construct-tree`( $p, \mathbf{r}, \ell$ )
-



■ **Figure 2** An example for the construction of  $T^\circ$ . The tree on the left side is  $\mathbf{T}$ , and the tree on the right side is  $T^\circ$ . The labels of the nodes in  $\mathbf{T}$  are shown besides them. In  $T^\circ$ , selectors, copiers and leaves are denoted as empty circles, solid circles and empty squares respectively. The nodes in the two yellow polygons are associated with  $r$  and  $a$  respectively. The numbers in the circles and squares indicate the labels associated with the nodes. In the example, the triples in  $\Gamma_r$  with the first coordinate being 1 are  $(1, 4, 6)$ ,  $(1, 5, 6)$  and  $(1, 5, 7)$ . The triples in  $\Gamma_a$  with the first coordinate being 5 are  $(5, 9, 11)$ ,  $(5, 9, 12)$  and  $(5, 10, 11)$ .

■ **Algorithm 2**  $\text{construct-tree}(p, u, \ell)$ .  $\triangleright p \in V^\circ, u \in \mathbf{V}, \ell \in L_u$

- 
- 1: **if**  $u$  has no children **then return**  $\triangleright p$  is a leaf node.
  - 2: let  $p$  be a selector node, let  $v$  and  $v'$  be the two children of  $u$  in  $\mathbf{T}$
  - 3: **for every**  $\ell' \in L_v, \ell'' \in L_{v'}$  such that  $(\ell, \ell', \ell'') \in \Gamma_u$  **do**
  - 4:     create a child  $p'$  of  $p$ , associated with  $u$ , let  $p'$  be a copier node,
  - 5:     create two children  $q$  and  $q'$  of  $p'$ , associate  $q$  with node  $v$  and label  $\ell'$ , associate  $q'$  with node  $v'$  and label  $\ell''$
  - 6:     call  $\text{construct-tree}(q, v, \ell')$  and  $\text{construct-tree}(q', v', \ell'')$
- 

Now we can define consistent sub-trees of  $T^\circ$ :

► **Definition 11** (Consistent sub-trees). *Given a sub-tree  $T$  of  $T^\circ$  that contains  $r$ , we say  $T$  is consistent if the following conditions hold.*

- *Every selector node  $p$  in  $T$  has exactly one child in  $T$ .*
- *If  $p$  is a copier node in  $T$ , then both of its children in  $T^\circ$  are in  $T$ .*

The definition explains the names “selector” and “copier”: a selector node  $p$  in  $T$  needs to select one of its children in  $T^\circ$  and add it to  $T$ , and the children of a copier node  $p$  will follow the node  $p$  to enter  $T$ .

It is easy to see a one-to-one correspondence between consistent labelings  $\vec{\ell} = (\ell_u \in L_u)_{u \in \mathbf{V}}$  of  $\mathbf{T}$ , and consistent sub-trees  $T$  of  $T^\circ$ . Given the consistent labeling  $\vec{\ell}$ , the correspondent sub-tree  $T$  of  $T^\circ$  can be constructed as follows. First, we add  $r$  and its child  $p$  associated with label  $\ell_r$  to  $T$ . Then we grow the tree from  $p$  using a recursive procedure. Assume  $p$  is associated with node  $u$  in  $\mathbf{T}$  and label  $\ell \in L_u$ . If  $u$  is a leaf, we stop the procedure. Otherwise let  $v$  and  $v'$  be the two children of  $u$ , then we add the copier child  $p'$  of  $p$  that corresponds to the tuple  $(\ell_r, \ell_v, \ell_{v'})$  to  $T$ . We also add its two children  $q$  and  $q'$  to  $T$ . Then we run the procedure recursively over  $q$  and  $q'$ . Conversely, given a consistent sub-tree  $T$  of  $T^\circ$ , we can recover a consistent labeling  $\vec{\ell}$  of  $\mathbf{T}$ .

For convenience, we extend the costs  $(c_\ell^i)_{i \in [m], \ell \in L}$  to vertices in  $V^\circ$ : For every non-root selector node or leaf node  $p \in V^\circ$  associated with a label  $\ell$ , we define  $c_p^i = c_\ell^i$  for every  $i \in [m]$ . For the root or a copier node  $p$ , we define  $c_p^i = 0$ . For a consistent sub-tree  $T = (V, E)$  of  $T^\circ$ , and  $i \in [m]$ , we define its type- $i$  cost to be  $\text{cost}^i(T) = \sum_{p \in V} c_p^i$ . This will be the same as  $\text{cost}^i(\vec{\ell})$ , for the labeling  $\vec{\ell}$  correspondent to  $T$ .

We also extend the groups  $S_1, S_2, \dots, S_k$  to node sets in  $T^\circ$ : for every  $t \in [k]$ ,  $S'_t$  contains the set of nodes  $p \in V^\circ$  whose associated label is in  $S_t$ . Then, a consistent labeling  $\vec{\ell}$  covers a group  $S_t$  if and only if the correspondent sub-tree  $T = (V, E)$  covers  $S'_t$ , namely,  $V \cap S'_t \neq \emptyset$ .

Therefore, we are guaranteed that there is a consistent sub-tree  $T^*$  of  $T^\circ$  that covers all groups  $S'_1, S'_2, \dots, S'_k$ , and has  $\text{cost}^i(T^*) \leq 1$  for every  $i \in [m]$ . Our goal is to output a random consistent sub-tree  $T$  satisfying the conditions correspondent to a and b. This is done using an LP-based algorithm.

### 3.2 The LP relaxation for finding $T = (V, E)$

Now we describe the LP relaxation that we use to find  $T = (V, E)$ . For every vertex  $p \in V^\circ$ , we use  $x_p$  to indicate if  $p$  is in  $T$ , i.e.,  $p \in V$ . For every  $t \in [k]$  and  $q \in S'_t$ , we use  $y_q^t$  to indicate if  $q$  is the node in  $T$  we choose to cover  $S'_t$ . There might be multiple nodes in  $V \cap S'_t$ , and in this case, we only choose one node in the set to cover  $S'_t$ ; the choice can be arbitrary. The LP is as follows.

$$\begin{aligned}
 x_r &= 1 & (2) \quad & \sum_{q \in \Lambda^*(p) \cap S'_t} y_q^t \leq x_p, \forall p \in V^\circ, t \in [k] & (6) \\
 \sum_{q \in \Lambda(p)} x_q &= x_p, \forall \text{ selector } p \in V^\circ & (3) \quad & \sum_{q \in S'_t} y_q^t = 1, \forall t \in [k] & (7) \\
 x_q &= x_p, \forall \text{ copier } p \in V^\circ, q \in \Lambda(p) & (4) \quad & \sum_{q \in \Lambda^*(p)} c_q^i x_q \leq x_p, \forall p \in V^\circ, i \in [m] & (8) \\
 x_p &\geq 0, \forall p \in V^\circ & (5) \quad & & 
 \end{aligned}$$

Constraints (2)-(5) in the LP are for the consistency requirements. (2) says the root is always in  $T$ . (3) says if a selector node  $p$  is in  $T$ , then exactly one of its children is in  $T$ . (4) says if a copier node  $p$  is in  $T$ , and  $q$  is a child of  $p$ , then  $q$  is also in  $T$ . (5) is the non-negativity condition. (6) and (7) deal with the covering requirements. (6) says if  $p$  is in  $T$ , then we choose at most one descendant of  $p$  to cover the group  $S'_t$ ; notice that the constraint implies  $y_p^t \leq x_p$  if  $p \in S'_t$ . (7) says we choose exactly one node in  $T$  to cover  $S'_t$ . (8) handles the cost requirement: If  $p$  is included in  $T$ , then the type- $i$  cost of the descendants of  $p$  in  $T$  is at most 1.

### 3.3 The rounding algorithm

The rounding algorithm is based on [9]. We solve LP(2) to obtain a solution  $x \in [0, 1]^{V^\circ}$ . We add  $r$  to  $T$  and call `recursive-rounding`( $r$ ) to obtain a sub-tree  $T = (V, E)$ . The procedure is defined in Algorithm 3.

► **Observation 12.**  *$T$  is always consistent. For every  $p \in V^\circ$ , we have  $\Pr[p \in V] = x_p$ .*

**Proof.** For a selector node  $p$  in  $T$ , we always choose exactly one child of  $p$  and add it to  $T$ . For a copier node  $p$  added to  $T$  and one of its child  $q$ ,  $q$  is added to  $T$  with probability 1. By the probabilities we add nodes to  $T$ , we can see that  $\Pr[p \in V] = x_p$  for every  $p \in V^\circ$ . ◀

■ **Algorithm 3** recursive-rounding( $p$ ).

---

```

1: if  $p$  is a selector node then
2:   choose one vertex  $q \in \Lambda(p)$  randomly, so that  $q$  is chosen with probability  $\frac{x_q}{x_p}$ 
3:   add  $q$  to  $T$ , and call recursive-rounding( $q$ )
4: else ▷  $p$  is a copier or leaf node
5:   for every  $q \in \Lambda(p)$  do
6:     with probability  $\frac{x_q}{x_p} = 1$ : add  $q$  to  $T$ , and call recursive-rounding( $q$ )
    
```

---

**4** Analysis of probabilities of group coverage

We fix  $t \in [k]$  and analyze the probability that  $T$  covers the group  $S'_t$ ; or equivalently, the correspondent labeling covers the group  $S_t$ . This will prove Property a. For every vertex  $p \in V^\circ$ , we define  $z_p := \sum_{q \in \Lambda^*(p) \cap S'_t} y_q^t$ , which indicates whether  $S'_t$  is covered by vertices in the sub-tree rooted at  $p$ . By (6), we have  $z_p \leq x_p$ . By (7), we have  $z_r = 1 = x_r$ .

We define the height of a node  $p \in V^\circ$  to be the maximum number of copier nodes in a path from  $p$  to one of its descendant leaves. We prove in Appendix A the following lemma:

► **Lemma 13.** *Assume  $p \in V^\circ$  has height  $h$ . Then  $\Pr \left[ \Lambda^*(p) \cap V \cap S'_t \neq \emptyset \mid p \in V \right] \geq \frac{1}{h+1} \frac{z_p}{x_p}$ .*

Notice that the height of the root  $r$  of  $T^\circ$  is  $D - 1$ . Applying the above lemma with  $p = r$ , we have that  $T$  covers group  $S'_t$  with probability at least  $\frac{1}{D} \cdot \frac{z_r}{x_r} = \frac{1}{D}$ . So, the correspondent  $\vec{\ell}$  covers  $S_t$  with probability at least  $\frac{1}{D}$ , proving Property a.

### 4.1 Concentration bound on costs

In this section, we prove Property b. We fix an index  $i \in [m]$  and analyze the type- $i$  cost of  $T = (V, E)$ . For notation convenience, we use  $c_p$  to denote  $c_p^i$ , and cost for type- $i$  cost.

For every vertex  $p \in V^\circ$ , let  $w_p = \sum_{q \in \Lambda^*(p)} c_q x_q$  be the fractional cost incurred by the sub-tree of  $T^\circ$  rooted at  $p$ . By (8), we have  $w_p \leq x_p$ . Let  $W_p = \sum_{q \in \Lambda^*(p) \cap V} c_q$  be the cost of  $T$  incurred by descendants of  $p$ . So, we have  $\mathbb{E}[W_p] = w_p$ .

As is typical, we shall introduce a parameter  $s > 0$  and consider the expectation of the random exponential variables  $e^{sW_p}$ . Later we shall set  $s = \ln(1 + \frac{1}{2D})$ , but the main lemma holds for any  $s > 0$ . We define an  $\alpha_h$  for every integer  $h \geq 0$  as  $\alpha_0 = e^s$  and  $\alpha_h = e^{\alpha_{h-1}-1}, \forall h \geq 1$ . Notice that  $\alpha_0, \alpha_1, \dots$  is an increasing sequence.

In this section, we count selector nodes in the definition of heights: the height of a node  $p \in V^\circ$  is the maximum number of selector nodes in a path from  $p$  to its descendant leaf.

► **Lemma 14.** *For any node  $p$  in  $T^\circ$  of height  $h$ , we have  $\mathbb{E} \left[ e^{sW_p} \mid p \in V \right] \leq \alpha_h^{w_p/x_p}$ .*

**Proof.** We prove the lemma for nodes  $p$  from bottom to top of the tree  $T^\circ$ . Focus on a node  $p$  of height  $h$ . If  $p$  is a copier or leaf node, all children of  $p$  has height at most  $h$ . We have

$$\begin{aligned} \mathbb{E} \left[ e^{sW_p} \mid p \in V \right] &= e^{sc_p} \prod_{q \in \Lambda(p)} \mathbb{E} \left[ e^{sW_q} \mid q \in V \right] \\ &= \alpha_0^{c_p x_p / x_p} \prod_{q \in \Lambda(p)} \alpha_h^{w_q / x_p} \leq \alpha_h^{c_p x_p / x_p} \prod_{q \in \Lambda(p)} \alpha_h^{w_q / x_p} = \alpha_h^{w_p / x_p}. \end{aligned}$$

The last inequality used that  $\alpha_0 \leq \alpha_h$ , and the last equality used that  $w_p = c_p x_p + \sum_{q \in \Lambda(p)} w_q$ .

Now suppose  $p$  is a selector. Then all children of  $p$  have height at most  $h-1$ . Conditioned on  $p \in V$ , the rounding procedure adds exactly one child  $q$  of  $p$  to  $V$ . Then, we have

$$\begin{aligned} \mathbb{E} \left[ \mathbf{e}^{sW_p} \mid p \in V \right] &= \mathbf{e}^{sc_p} \cdot \sum_{q \in \Lambda(p)} \frac{x_q}{x_p} \mathbb{E} \left[ \mathbf{e}^{sW_q} \mid q \in V \right] = \mathbf{e}^{sc_p} \cdot \sum_{q \in \Lambda(p)} \frac{x_q}{x_p} \alpha_{h-1}^{w_q/x_q} \\ &\leq \mathbf{e}^{sc_p} \left( \left( \frac{w_p}{x_p} - c_p \right) \cdot \alpha_{h-1} + \left( 1 - \frac{w_p}{x_p} + c_p \right) \right) = \mathbf{e}^{sc_p} \left( 1 + \left( \frac{w_q}{x_q} - c_p \right) (\alpha_{h-1} - 1) \right) \\ &\leq \mathbf{e}^{sc_p} \cdot \exp \left( \left( \frac{w_p}{x_p} - c_p \right) (\alpha_{h-1} - 1) \right) = \mathbf{e}^{sc_p} \cdot \alpha_h^{w_p/x_p - c_p} \leq \alpha_h^{w_p/x_p}. \end{aligned}$$

To see the inequality in the second line, we notice the following four facts: (i)  $\alpha_{h-1}^\theta$  is a convex function of  $\theta$ , (ii)  $w_q/x_q \in [0, 1]$  for every  $q \in \Lambda(p)$ , (iii)  $\sum_{q \in \Lambda(p)} \frac{x_q}{x_p} = 1$  and (iv)  $\sum_{q \in \Lambda(p)} \frac{x_q}{x_p} \cdot \frac{w_q}{x_q} = \sum_{q \in \Lambda(p)} \frac{w_q}{x_p} = \frac{w_p}{x_p} - c_p$ . The equality in the last line is by the definition of  $\alpha_h$ . The last inequality used that  $\mathbf{e}^s = \alpha_0 \leq \alpha_h$ .  $\blacktriangleleft$

The height of the root  $r$  is  $D$ .<sup>3</sup> Now, we set  $s = \ln(1 + \frac{1}{2D})$ .

► **Lemma 15.** *For every  $h \in [0, D]$ , we have  $\alpha_h \leq 1 + \frac{1}{2D-h}$ .*

**Proof.** By definition,  $\alpha_0 = \mathbf{e}^s = 1 + \frac{1}{2D}$  and thus the statement holds for  $h = 0$ . Let  $h \in [1, D]$  and assume the statement holds for  $h-1$ . Then, we have

$$\begin{aligned} \alpha_h &= \mathbf{e}^{\alpha_{h-1}-1} \leq \mathbf{e}^{\frac{1}{2D-h+1}} \leq 1 + \frac{1}{2D-h+1} + \left( \frac{1}{2D-h+1} \right)^2 \\ &= 1 + \frac{2D-h+2}{(2D-h+1)^2} \leq 1 + \frac{1}{2D-h}. \end{aligned}$$

The first inequality used the induction hypothesis and the second one used that for every  $\theta \in [0, 1]$ , we have  $e^\theta \leq 1 + \theta + \theta^2$ .  $\blacktriangleleft$

To wrap up, we apply Lemma 14 on  $p = r$ . Notice that  $r \in V$  always happens, at  $W_r = \text{cost}^i(T)$ . We have  $\mathbb{E} \left[ \exp \left( \ln \left( 1 + \frac{1}{2D} \right) \cdot \text{cost}^i(T) \right) \right] \leq \alpha_D^{w_r/x_r} \leq 1 + \frac{1}{D}$  by Lemma 15 and that  $w_r \leq x_r = 1$ . Using the correspondence between sub-trees of  $T^\circ$  and labelings of  $\mathbf{T}$  proves Property b.

---

## References

- 1 B. Awerbuch, Y. Azar, E. Grove, M. Kao, P. Krishnan, and J. Vitter. Load balancing in the  $l_p$  norm. In *FOCS*, pages 383–391, 1995.
- 2 Y. Azar and S. Taub. All-norm approximation for scheduling on identical machines. In T. Hagerup and J. Katajainen, editors, *SWAT*, volume 3111, pages 298–310, 2004.
- 3 Hans L. Bodlaender. Nc-algorithms for graphs with small treewidth. In Jan van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science, 14th International Workshop, WG '88, Amsterdam, The Netherlands, June 15-17, 1988, Proceedings*, volume 344 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1988. doi:10.1007/3-540-50728-0\_32.
- 4 P. Chalermsook, S. Das, B. Laekhanukit, and D. Vaz. Beyond metric embedding: Approximating group steiner trees on bounded treewidth graphs. In *SODA*, pages 737–751, 2017.

---

<sup>3</sup> The height of  $r$  is  $D+1$  by definition, but Lemma 14 holds when we define its height to be  $D$ , as one can collapse the first two levels of  $T^\circ$  into one level.

- 5 Eden Chlamtác, Michael Dinitz, and Thomas Robinson. Approximating the Norms of Graph Spanners. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*, volume 145 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:22, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.11.
- 6 Eden Chlamtác, Michael Dinitz, and Thomas Robinson. The Norms of Graph Spanners. 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 40:1–40:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2019.40.
- 7 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, pages 624–633, 2014.
- 8 M. Furer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994.
- 9 N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.
- 10 D. Golovin, A. Gupta, A. Kumar, and K. Tangwongsan. All-norms and all- $l_p$ -norms approximation algorithms. In *IACS*, volume 2 of *LIPIcs*, pages 199–210, 2008.
- 11 Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li.  $o(\log^2 k / \log \log k)$ -approximation algorithm for directed steiner tree: A tight quasi-polynomial time algorithm. *SIAM Journal on Computing*, 52(2):STOC19–298–STOC19–322, 2023. doi:10.1137/20M1312988.
- 12 X. Guo, G. Kortsarz, B. Laekhanukit, S. Li, D. Vaz, and J. Xian. On approximating degree-bounded network design problems. *Algorithmica*, 84(5):1252–1278, 2022.
- 13 Mohammad Taghi Hajiaghayi. A list of open problems in bounded degree network design. *The 8th Workshop on Flexible Network Design*, 2016.
- 14 E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *STOC*, pages 585–594, 2003.
- 15 Sungjin Im and Shi Li. Improved approximations for unrelated machine scheduling. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2917–2946, 2023. doi:10.1137/1.9781611977554.ch111.
- 16 K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
- 17 Guy Kortsarz and Zeev Nutov. The minimum degree group steiner problem. *Discret. Appl. Math.*, 309:229–239, 2022.
- 18 V. Kumar, M. Marathe, S. Parthasarathy, and A. Srinivasan. A unified approach to scheduling on unrelated parallel machines. *J. ACM*, 56(5):28:1–28:31, 2009.
- 19 C. Lau L, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, 2011.
- 20 L. C. Lau, J. Naor, M. R. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. *SIAM J. Comput.*, 39(3):1062–1087, 2009.
- 21 Shi Li, Chenyang Xu, and Ruilong Zhang. Polylogarithmic Approximation for Robust s-t Path. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, 2024.
- 22 Manmeet Kaur Nisha Sharma. A survey of vlsi techniques for power optimization and estimation of optimization. *International Journal of Emerging Technology and Advanced Engineering*, 4, 2014.
- 23 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *J. ACM*, 62(1), March 2015. doi:10.1145/2629366.
- 24 Y. Wang, X. Hong, T. Jing, Y. Yang, X. Hu, and Guiying Yan. An efficient low-degree RMST algorithm for VLSI/ULSI physical design. In *PATMOS*, pages 442–452, 2004.



## A

 Proof of Lemma 13

We repeat the lemma below:

► **Lemma 13.** *Assume  $p \in V^\circ$  has height  $h$ . Then  $\Pr \left[ \Lambda^*(p) \cap V \cap S'_t \neq \emptyset \mid p \in V \right] \geq \frac{1}{h+1} \cdot \frac{z_p}{x_p}$ .*

**Proof.** If  $p \in S'_t$  then  $\Pr \left[ \Lambda^*(p) \cap V \cap S'_t \neq \emptyset \mid p \in V \right] = 1 \geq \frac{z_p}{x_p}$ . The inequality holds trivially. So, we can assume  $p \notin S'_t$ , and we prove the lemma for nodes  $p$  from bottom to top in the tree  $T^\circ$ . Suppose  $p$  is a leaf; then  $h = 0$ , and  $z_p = 0$  as we assumed  $p \notin S'_t$ . The inequality trivially holds.

So we can assume  $p$  to be a non-leaf node of height  $h$  and assume the lemma holds for every  $q \in \Lambda(p)$ . First, assume  $p$  is a selector node. Then all children of  $p$  have height at most  $h$ .

$$\Pr \left[ \Lambda^*(p) \cap V \cap S'_t \neq \emptyset \mid p \in V \right] \geq \sum_{q \in \Lambda(p)} \frac{x_q}{x_p} \cdot \frac{1}{h+1} \cdot \frac{z_q}{x_q} \geq \sum_{q \in \Lambda(p)} \frac{1}{h+1} \cdot \frac{z_q}{x_p} = \frac{1}{h+1} \cdot \frac{z_p}{x_p}.$$

Then consider the case that  $p$  is a copier node. All children of  $p$  have height at most  $h-1$ . Even though  $p$  has exactly two children, our analysis works if it has any number of children.

$$\begin{aligned} \Pr \left[ \Lambda^*(p) \cap V \cap S'_t \neq \emptyset \mid p \in V \right] &\geq 1 - \prod_{q \in \Lambda(p)} \left( 1 - \frac{1}{h} \cdot \frac{z_q}{x_q} \right) = 1 - \prod_{q \in \Lambda(p)} \exp \left( -\frac{1}{h} \cdot \frac{z_q}{x_p} \right) \\ &= 1 - \exp \left( -\frac{1}{h} \cdot \frac{z_p}{x_p} \right) \geq \frac{1}{h} \cdot \frac{z_p}{x_p} - \frac{1}{2} \left( \frac{1}{h} \cdot \frac{z_p}{x_p} \right)^2 \geq \frac{1}{h} \cdot \frac{z_p}{x_p} - \frac{1}{2} \left( \frac{1}{h} \right)^2 \frac{z_p}{x_p} \\ &= \left( \frac{2h-1}{2h^2} \right) \frac{z_p}{x_p} \geq \frac{1}{h+1} \cdot \frac{z_p}{x_p}. \end{aligned}$$

The first equality in the first line used that  $x_q = x_p$  for every  $q \in \Lambda(p)$ . The second equality used that  $z_p = \sum_{q \in \Lambda(p)} z_q$  as  $p \notin S'_t$ . The first inequality in the second line used that  $e^{-\theta} \leq 1 - \theta + \frac{\theta^2}{2}$  for every  $\theta \geq 0$ . The second inequality used that  $\frac{z_p}{x_p} \leq 1$ . ◀

## B

 Reduction of Degree-Bounded Group Steiner Tree on Bounded-Treewidth Graphs to Tree-Labeling Problem

In this section, we prove Theorem 2, by reducing Group Steiner Tree with degree bounds on bounded treewidth graphs to the tree labeling problem studied in Section 3. Recall the input of the problem contains a graph  $G = (V, E)$  with edge costs  $c \in R_{\geq 0}^E$ , a root  $r$ ,  $k$  groups  $S_1, S_2, \dots, S_k$  of vertices, and a degree bound  $\text{db}_v \in \mathbb{Z}_{>0}$  for every  $v \in V$ . Without loss of generality, we assume  $\{r\}, S_1, S_2, \dots, S_k$  are mutually disjoint. Again, we use  $\text{opt}$  to denote the minimum cost of a valid subgraph  $H$ .

Let  $\mathbf{T} = (B, \mathbf{E})$  be the tree decomposition of the graph  $G = (V, E)$ . Every  $b \in B$  is called a bag and let  $X_b \subseteq V$  be the set of vertices contained in the bag  $b$ . We can add the root  $r$  to all the bags, which increases the maximum size of a bag by at most 1. It was shown in [3] that we can assume  $\mathbf{T}$  is a rooted binary tree of depth  $O(\log n)$ , by sacrificing the bag size by an  $O(1)$  factor. We summarize the properties as follows:

- $\mathbf{T}$  is a full binary tree rooted at  $\mathbf{r}$ , with depth  $O(\log n)$ .
- $|X_b| \leq O(1) \cdot \text{tw}$  for every  $b \in B$ .
- For every edge  $(u, v) \in E$ , there is some  $b \in B$  with  $\{u, v\} \subseteq X_b$ .
- For every  $v \in V$ , the set of bags  $b$  with  $v \in X_b$  is connected in  $\mathbf{T}$ .

For every  $e \in E$ , let  $b_e$  be the highest node  $b$  such that  $X_b$  contains both end vertices of  $e$ . This is well-defined due to the last property in the above list. For every  $b \in B$ , we let  $E_b = \{e \in E : b_e = b\}$ . So,  $(E_b)_{b \in B}$  forms a partition of  $E$ .

**Notations on Partitions.** Given two partitions  $\Pi$  and  $\Pi'$  of a common set  $X$ , we say  $\Pi'$  refines  $\Pi$  if any two elements in  $X$  that are in the same set in  $\Pi'$  are also in the same set in  $\Pi$ . We use  $\Pi' \leq \Pi$  to denote that  $\Pi'$  refines  $\Pi$ . Given two partitions  $\Pi$  and  $\Pi'$  of  $X$ , we use  $\Pi \vee \Pi'$  to denote the join of  $\Pi$  and  $\Pi'$  w.r.t the relation  $\leq$ . That is, we define a graph where there is an edge between  $u$  and  $v$  if they are in the same set in  $\Pi$  or  $\Pi'$ . Then two vertices  $u$  and  $v$  are in the same set in the partition  $\Pi \vee \Pi'$  if and only if they are in the same connected component in the graph.

Abusing notations slightly, if an element  $v$  is not included in a partition  $\Pi$ , we treat  $\{v\}$  as a singleton set in  $\Pi$ . This allows us to extend the operators  $\leq$  and  $\vee$  to two partitions  $\Pi$  and  $\Pi'$  with different ground sets. Given a partition  $\Pi$  and a set  $X$ , we let  $\Pi[X]$  be the partition  $\Pi$  restricted to the ground set  $X$ : two elements  $u, v \in X$  are in the same set in  $\Pi[X]$  if and only if they are in the same set in  $\Pi$ .

For any set  $F \subseteq E$  of edges, we define  $CC(F)$  to be the partition of the vertices incident to  $F$ , such that  $u$  and  $v$  are in the same set in  $CC(F)$  if and only if they are in the same connected component in  $(V, F)$ .

**Construction of Labels and Consistency Triples.** The tree  $\mathbf{T}$  for the tree-labeling instance is the same as the decomposition tree  $\mathbf{T}$ . (This is the reason we use the same notion  $\mathbf{T}$ .) So we have  $\mathbf{V} = B$ . Now we fix a bag  $b \in B$  and define the set  $L_b$  of labels for  $b$ . To define the labels, we let  $H = (V_H, E_H)$  be any sub-graph of  $G$ , which we should think of as the output of the GST problem. Fix a bag  $b \in B$ , let  $\Lambda^*(b)$  be the set of descendants of  $b$  in  $\mathbf{T}$ , including  $b$  itself. We then make the following definitions:

- $F_b(H) := E_H \cap E_b$  is the set of edges from  $E_b$  that are included in  $H$ .
- $\Pi_b^\downarrow(H)$  is the partition of  $X_b$  so that two vertices  $u, v \in X_b$  is in the set in  $\Pi_b^\downarrow(H)$  if and only if they are connected in the graph  $(V_H, E_H \cap \bigcup_{b' \in \Lambda^*(b)} E_{b'})$ .
- $\Pi_b^\uparrow(H)$  is the partition of  $X_b$  so that two vertices  $u, v \in X_b$  is in the set in  $\Pi_b^\uparrow(H)$  if and only if they are connected in the graph  $(V_H, E_H \cap \bigcup_{b' \in B \setminus \Lambda^*(b) \cup \{b\}} E_{b'})$ .

In words,  $\Pi_b^\downarrow(H)$  and  $\Pi_b^\uparrow(H)$  respectively indicate the partition of  $X_b$  correspondent to the edges of  $H$  in bags below and above  $b$  respectively.

Without knowing  $H$ , we can define the label set  $L_b$  for  $b$  to be all tuples  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)$  such that  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow) = (F_b(H), \Pi_b^\downarrow(H), \Pi_b^\uparrow(H))$  for some valid output graph  $H$ . We then define the consistency tuples  $\Gamma_b$ 's so that a consistent labeling gives a valid outputs sub-graph  $H$ .

Formally, let  $L_b$  be the set of all tuples  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)$  such that

- $F_b \subseteq E_b$  is a forest over  $X_b$ ,  $CC(F_b) \leq \Pi_b^\downarrow$  and  $CC(F_b) \leq \Pi_b^\uparrow$ ,
- if  $b = \mathbf{r}$ , then  $\Pi_b^\uparrow = CC(F_b)$ , and
- if  $b$  is a leaf, then  $\Pi_b^\downarrow = CC(F_b)$ .

Then we define the set  $\Gamma_b$  of triples, for an inner vertex  $b$  in  $\mathbf{T}$  with two children  $b'$  and  $b''$ . We have  $((F_b, \Pi_b^\downarrow, \Pi_b^\uparrow), (F_{b'}, \Pi_{b'}^\downarrow, \Pi_{b'}^\uparrow), (F_{b''}, \Pi_{b''}^\downarrow, \Pi_{b''}^\uparrow)) \in \Gamma_b$  if and only if

- $\Pi_b^\downarrow = (\Pi_{b'}^\downarrow \vee \Pi_{b''}^\downarrow \vee CC(F_b))[X_b]$ ,
- $\Pi_{b'}^\uparrow = (\Pi_b^\uparrow \vee \Pi_{b''}^\downarrow \vee CC(F_{b'}))[X_{b'}]$ , and
- $\Pi_{b''}^\uparrow = (\Pi_b^\uparrow \vee \Pi_{b'}^\downarrow \vee CC(F_{b''}))[X_{b''}]$ .

▷ **Claim 16.** Let  $\{(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)\}_{b \in B}$  be a consistent labeling of the tree  $\mathbf{T}$ . Let  $H = (V, \bigcup_{b \in B} F_b)$ . Then we have  $\Pi_b^\downarrow[H] = \Pi_b^\downarrow$  and  $\Pi_b^\uparrow[H] = \Pi_b^\uparrow$  for every  $b \in B$ .

The claim says that if the labels are consistent, then  $\Pi_b^\downarrow$  and  $\Pi_b^\uparrow$  represent their true values.

**Construction of Covering and Cost Constraints.** The requirement that all groups are connected to  $r$  can be captured by the covering constraint in the tree-labeling problem. For every  $t \in [k]$ , a label  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow) \in L_b$  for some  $b \in B$  can satisfy the group  $S_t$  if for some  $s \in S_t$  we have  $(s, r)$  are in the same set in the partition  $\Pi_b^\downarrow \vee \Pi_b^\uparrow$ .

The edge costs and degree constraints can be captured by the cost constraints in the tree-labeling instance. Consider the costs first. Using binary search, we assume we know the optimum cost  $C^*$  for the instance. For every bag  $b \in B$  and every label  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)$ , the cost of the label is  $c(F_b) := \sum_{e \in F_b} c_e$ . We disallow this label by removing it if  $c(F_b) > C^*$ . Scaling all costs by  $C^*$  so that all costs are in  $[0, 1]$ . So, the cost being at most  $C^*$  in the group Steiner tree instance is equivalent to that the cost of all labels is at most 1.

Finally, we consider the degree constraints  $d_H(v) \leq \text{db}_v$  for every  $v \in V$ . For every  $v \in V$ , we define a cost constraint in the tree-labeling instance. For every bag  $b \in B$  with  $v \in X_b$ , and every label  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)$ , the cost of the label is  $|\delta(v) \cap F_b|$ , where  $\delta(v)$  is the incident edges of  $v$  in  $G$ . Again, we disallow the label if  $|\delta(v) \cap F_b| > \text{db}_v$ , and we scale the costs by  $\text{db}_v$  so that all costs are in  $[0, 1]$ . Then the degree constraint on  $v$  is reduced to this cost requirement in the tree labeling instance.

**Wrapping Up.** We then run the algorithm in Theorem 10 on the constructed tree-labeling instance. Let  $(F_b, \Pi_b^\downarrow, \Pi_b^\uparrow)$  be the label of a bag  $b$ , and let  $H = (V, \bigcup_{b \in B} F_b)$ . By Claim 16, the consistency constraints guarantee that the  $\Pi_b^\downarrow$  and  $\Pi_b^\uparrow$  truthfully represent the connectivity of the graph  $G$ . So, if the covering constraint for a group  $S_t$  is satisfied, then  $H$  indeed connects  $r$  and  $S_t$ . Recall that  $D = O(\log n)$  is the depth of the tree  $\mathbf{T}$ . By Properties a and b, we have



- For every  $t \in [k]$ ,  $H$  connects  $r$  and  $S_t$  with probability at least  $\frac{1}{D}$ .
- $\mathbb{E} \left[ \exp(\ln(1 + \frac{1}{2D}) \cdot \frac{c(H)}{C^*}) \right] \leq 1 + \frac{1}{D}$ .
- $\mathbb{E} \left[ \exp(\ln(1 + \frac{1}{2D}) \cdot \frac{d_H(v)}{\text{db}_v}) \right] \leq 1 + \frac{1}{D}$  for every  $v \in V$ .

We run the algorithm for  $M = \Theta(D \log n) = \Theta(\log^2 n)$  times, with a large hidden constant in the  $O(\cdot)$  notation, and output the union  $H$  of all sub-graphs constructed by the  $M$  times. With high probability, all groups are connected to  $r$  in  $H$ .  $\mathbb{E}[\exp(\ln(1 + \frac{1}{2D}) \cdot \frac{d_H(v)}{\text{db}_v})] \leq (1 + \frac{1}{D})^M = n^{O(1)}$ . Using Markov inequality, we have  $\exp(\ln(1 + \frac{1}{2D}) \cdot \frac{d_H(v)}{\text{db}_v}) \leq n^{O(1)}$  for every  $v \in V$  with high probability. That is,  $d_H(v) \leq O(\text{db}_v \log n \cdot D) = O(\log^2 n) \text{db}_v$  with high probability. Similarly, with high probability, we have  $c(H) \leq O(\log^2 n) C^*$ .

We then analyze the running time of the algorithm. The key parameter deciding the running time is  $\Delta$ , the maximum size of a label set  $L_b$ . As we assumed  $F_b$  is a forest over  $X_b$  and  $|X_b| \leq O(\text{tw})$ , there are  $\text{tw}^{O(\text{tw})}$  different possibilities for  $F_b$ . There are also  $\text{tw}^{O(\text{tw})}$  possibilities for each of  $\Pi_b^\downarrow$  and  $\Pi_b^\uparrow$ . So,  $|L_b| \leq \text{tw}^{O(\text{tw})}$  for every  $b \in B$ . Therefore, the running time of the algorithm is  $\text{poly}(n) \cdot \Delta^{O(D)} = \text{poly}(n) \cdot (\text{tw}^{O(\text{tw})})^{O(\log n)} = n^{O(\text{tw} \log \text{tw})}$ . This finishes the proof of Theorem 2.



# Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center

Fedor V. Fomin  

University of Bergen, Norway

Petr A. Golovach  

University of Bergen, Norway

Tanmay Inamdar  

Indian Institute of Technology Jodhpur, Jodhpur, India

Saket Saurabh  

The Institute of Mathematical Sciences, HBNI, Chennai, India

University of Bergen, Norway

Meirav Zehavi  

Ben-Gurion University of the Negev, Beer-Sheva, Israel

---

## Abstract

We propose a novel clustering model encompassing two well-known clustering models:  $k$ -center clustering and  $k$ -median clustering. In the HYBRID  $k$ -CLUSTERING problem, given a set  $P$  of points in  $\mathbb{R}^d$ , an integer  $k$ , and a non-negative real  $r$ , our objective is to position  $k$  closed balls of radius  $r$  to minimize the sum of distances from points not covered by the balls to their closest balls. Equivalently, we seek an optimal  $L_1$ -fitting of a union of  $k$  balls of radius  $r$  to a set of points in the Euclidean space. When  $r = 0$ , this corresponds to  $k$ -MEDIAN; when the minimum sum is zero, indicating complete coverage of all points, it is  $k$ -CENTER.

Our primary result is a bicriteria approximation algorithm that, for a given  $\varepsilon > 0$ , produces a hybrid  $k$ -clustering with balls of radius  $(1 + \varepsilon)r$ . This algorithm achieves a cost at most  $1 + \varepsilon$  of the optimum, and it operates in time  $2^{(kd/\varepsilon)^{O(1)}} \cdot n^{O(1)}$ . Notably, considering the established lower bounds on  $k$ -center and  $k$ -median, our bicriteria approximation stands as the best possible result for HYBRID  $k$ -CLUSTERING.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Facility location and clustering; Theory of computation  $\rightarrow$  Fixed parameter tractability

**Keywords and phrases** clustering,  $k$ -center,  $k$ -median, Euclidean space, fpt approximation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.4

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.08295> [17]

**Funding** *Fedor V. Fomin:* Supported by the Research Council of Norway via the project BWCA (grant no. 314528).

*Petr A. Golovach:* Supported by the Research Council of Norway via the project BWCA (grant no. 314528).

*Saket Saurabh:* The author is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416); and he also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

*Meirav Zehavi:* The research was supported by the European Research Council (ERC) grant no. 101039913 (PARAPATH).



© Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Saket Saurabh, and Meirav Zehavi; licensed under Creative Commons License CC-BY 4.0

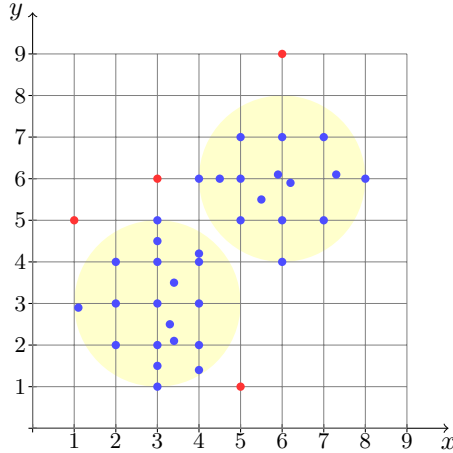
Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 4; pp. 4:1–4:19



Leibniz International Proceedings in Informatics

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



■ **Figure 1** Two disks of radius 2 cover all except four points that are colored red. The total sum of distances from these points to the yellow disks is  $2(1 + \sqrt{8} - 2)$ .

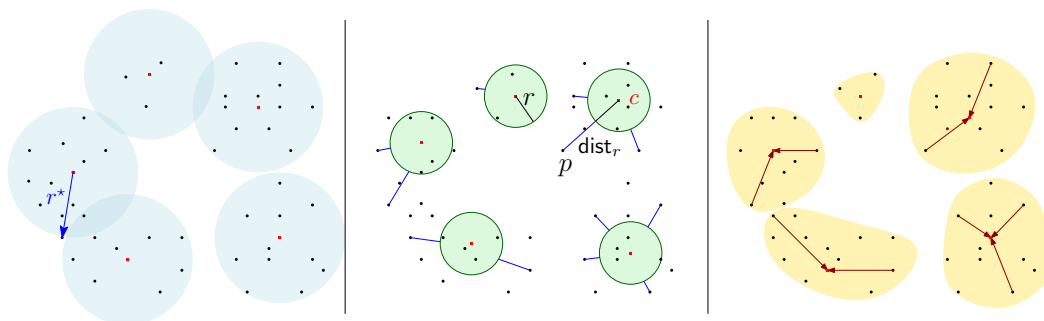
### 1 Introduction

Suppose we want to install a set of  $k$  access points (APs) at certain locations to provide wireless internet (Wi-Fi) coverage to a group of people belonging to a certain area. Each AP is capable of providing Wi-Fi within a circular-shaped region (i.e., a *disk*) of fixed radius  $r$ , and it may not be possible to cover the entire region with  $k$  such disks. Thus, after placing  $k$  APs, some people may be *outliers*, that lie outside any of the  $k$  disks and do not receive Wi-Fi coverage. We can model this scenario as the classical  $k$ -CENTER WITH OUTLIERS problem, which is a crude model since it only cares about the *number* of outliers. However, our scenario is more nuanced. All people that lie within any of the  $k$  disks of radius  $r$  already receive Wi-Fi, whereas a person lying outside all of the  $k$  disks must travel to the boundary of the nearest disk in order to receive coverage. Naturally, we would like to minimize the total distance traveled by people. Motivated by this and several other problems in computational geometry/clustering, we consider the following clustering problem, which encompasses two fundamental variants of clustering:  $k$ -CENTER and  $k$ -MEDIAN. Given a set  $P$  of points in some metric space and integer  $k$  and real  $r \geq 0$ , our objective is to position  $k$  closed balls of radius  $r$  in a way that minimizes the sum of distances from points uncovered by the balls to their closest balls. In Figure 1, we provide an example of such clustering with  $k = 2$  and  $r = 2$ .

To define the new clustering formally, we need some definitions. We consider Euclidean inputs, i.e., all points belong to  $\mathbb{R}^d$  for some  $d \geq 1$  and the distance function  $\text{dist}(\cdot, \cdot)$  is given by the Euclidean ( $\ell_2$ ) distance. For a point  $p \in P$  and a finite set of points  $Q \subset \mathbb{R}^d$ , we define  $\text{dist}(p, Q) := \min_{q \in Q} \text{dist}(p, q)$ . Further, for  $x, y \in P$ , and a real  $r \geq 0$ , we define the shorthand  $\text{dist}_r(x, y) := \max \{ \text{dist}(x, y) - r, 0 \}$ .

|   |     |
|---|-----|
| HYBRID $k$ -CLUSTERING  |     |
| <b>Input.</b> A set $P \subset \mathbb{R}^d$ of $n$ points, an integer $k \geq 1$ , and a real $r \geq 0$ . |     |
| <b>Task.</b> Find a set $F \subset \mathbb{R}^d$ of size at most $k$ , that minimizes:                      |     |
| $\text{cost}_r(P, F) := \sum_{p \in P} \text{dist}_r(p, F)$   | (1) |

We denote an instance of HYBRID  $k$ -CLUSTERING as  $\mathcal{I} = (P, k, r, d)$ , where  $d$  denotes the dimension. When  $r = 0$ , the optimal cost of HYBRID  $k$ -CLUSTERING equals the optimal  $k$ -MEDIAN clustering cost of the instance. Thus in this case, HYBRID  $k$ -CLUSTERING reduces to  $k$ -MEDIAN. However, when  $r > 0$ ,  $\text{dist}_r(\cdot, \cdot)$  does not form a metric, and hence we cannot simply reduce the problem to  $k$ -MEDIAN. On the other hand, the minimum value  $r$  that guarantees the cost of HYBRID  $k$ -CLUSTERING to be zero is equal to the optimal  $k$ -CENTER value. In this sense, HYBRID  $k$ -CLUSTERING reduces to  $k$ -CENTER.



■ **Figure 2** Left:  $k$ -CENTER clustering, a special case of HYBRID  $k$ -CLUSTERING with  $r = r^*$ . All points are covered by  $k$  balls of radius  $r^*$  and  $\text{OPT}_{r^*} = 0$ . Right:  $k$ -MEDIAN clustering, a special case of HYBRID  $k$ -CLUSTERING with  $r = 0$ , and every point contributes its distance to the closest center (some are shown as brown arrows). Middle: A general instance of HYBRID  $k$ -CLUSTERING lies somewhere *in between* the two cases, where points outside radius- $r$  balls contribute the distance to the boundary (shown in blue).

## 1.1 Our Result and Techniques

The main result of this paper is a bicriteria approximation algorithm for HYBRID  $k$ -CLUSTERING. An  $\alpha$ -approximation to an instance  $\mathcal{I} = (P, k, r, d)$  is a subset  $F \subset \mathbb{R}^d$  of size  $k$  with  $\text{cost}_r(P, F) \leq \alpha \cdot \text{OPT}_r$ , where  $\text{OPT}_r := \text{cost}_r(P, F^*)$  denotes the cost of an optimal solution  $F^* \subset \mathbb{R}^d$  of size at most  $k$ . Furthermore, an  $(\alpha, \beta)$ -bicriteria approximation is a solution  $F \subset \mathbb{R}^d$  with  $\text{cost}_{\beta r}(P, F) \leq \alpha \cdot \text{OPT}_r$ . Here,  $\text{cost}_{\beta r}(P, F) = \sum_{p \in P} \text{dist}_{\beta r}(p, F)$ .

Consider the special case of  $r = r^*$ , where  $r^*$  is the optimal radius for  $k$ -CENTER. Then,  $\text{OPT}_{r^*} = 0$ . Therefore, a  $(\alpha, 1)$ -bicriteria approximation would return a solution of cost  $\alpha \cdot \text{OPT}_{r^*} = 0$  using radius  $1 \cdot r^*$ , i.e., an optimal solution for  $k$ -CENTER. On the other hand, a  $(1, \beta)$ -bicriteria approximation, for the special case of  $r = 0$ , would return an optimal-cost solution using the radius of  $\beta r = 0$ . That is, such an algorithm would optimally solve  $k$ -MEDIAN. Combining these observations with the established lower bounds from the literature for  $k$ -MEDIAN and  $k$ -CENTER in Euclidean spaces, implies the following bounds for HYBRID  $k$ -CLUSTERING.

► **Proposition 1.** *The following holds for HYBRID  $k$ -CLUSTERING even when the input is from  $\mathbb{R}^2$ .*

- *For any  $\alpha \geq 1$ , there exists no FPT in  $k$  algorithm that returns an  $(\alpha, 1)$ -approximation, unless  $\text{FPT} = \text{W}[1]$  [27].*
- *For any finite  $\beta \geq 1$ , there exists no polynomial-time algorithm that returns a  $(1, \beta)$ -approximation unless  $\text{P} = \text{NP}$  [29].*

*Further, assuming the Exponential-Time Hypothesis (ETH), if the input is from  $\mathbb{R}^d$  with  $d \geq 4$ , then there exists no  $n^{o(k)}$  time algorithm that returns a  $(1, \beta)$ -approximation, for any finite  $\beta \geq 1$  [10].*

#### 4:4 Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center

Given these results, a natural question arises: *Can we achieve a  $(1+\varepsilon, 1+\varepsilon)$ -approximation for HYBRID  $k$ -CLUSTERING, running in time  $f(k, \varepsilon) \cdot n^{\mathcal{O}(1)}$ , particularly in low-dimensional Euclidean spaces?* Our main theorem answers this question.

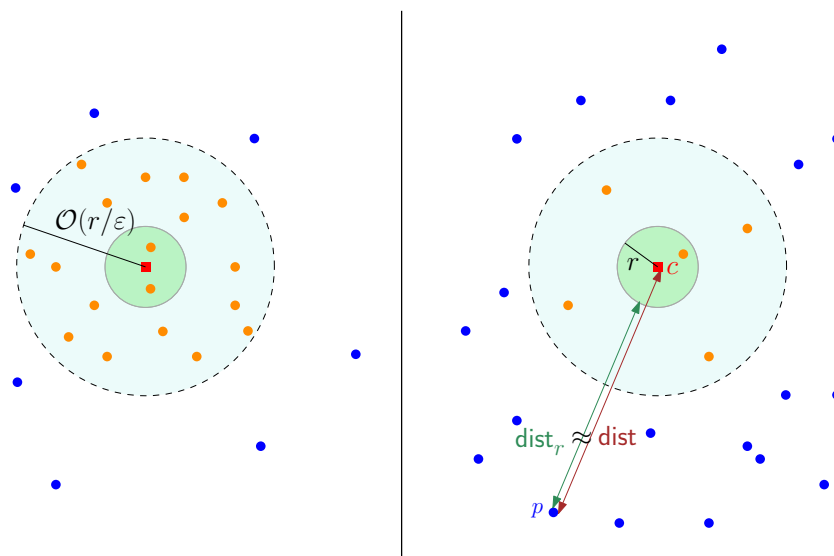
► **Theorem 2.** *Let  $0 < \varepsilon < 1$ . There exists a randomized algorithm that, given an instance of HYBRID  $k$ -CLUSTERING in  $\mathbb{R}^d$ , runs in time  $2^{\left(\frac{kd}{\varepsilon}\right)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ , and returns a  $(1 + \varepsilon, 1 + \varepsilon)$ -approximation with probability at least a positive constant.*

This randomized algorithm and the proof of correctness are described in Section 2. Here we discuss some of the main ideas. Recall that our objective, as the problem name suggests, is a “hybrid” of  $k$ -CENTER and  $k$ -MEDIAN. In our preprocessing steps, we first handle the inputs that behave *almost like* either of the two problems. Suppose we (approximately) know the optimal value of HYBRID  $k$ -CLUSTERING for the given set of points  $P$ , called  $\text{OPT}_r$ . First, in Lemma 4, if  $r > \text{OPT}_r$ , then we show that an approximate solution can be found using techniques used for approximating  $k$ -CENTER. Specifically, for each of the  $k$  centers in the optimal solution, we find a “nearby” center within distance  $\varepsilon r$  via overlaying a fine grid in the space. Thus, we can assume that  $r \leq \text{OPT}_r$ . Next, we consider the case when  $r$  is *too small* compared to  $\text{OPT}_r$ , namely, when  $r < \frac{\varepsilon \text{OPT}_r}{n}$ , and show that in this case, the input behaves like  $k$ -MEDIAN— an approximate  $k$ -MEDIAN solution is also an approximation for HYBRID  $k$ -CLUSTERING (Lemma 5). In this manner, we preprocess to handle inputs that resemble  $k$ -CENTER and  $k$ -MEDIAN, we obtain a relation between  $r$  and  $\text{OPT}_r$ , which can be used to discretize the distances, which can be used to bound the aspect ratio (i.e., the ratio of maximum to minimum positive distance) (Lemma 6).

After the preprocessing step, we obtain inputs that are not *immediately reducible* to  $k$ -CENTER/MEDIAN. To handle such inputs, we design an intricate recursive algorithm that, at each step, tries to simultaneously handle parts (i.e., clusters) of the input that can be handled by either of the two techniques. This algorithm is inspired by the sampling approach of Kumar, Sabharwal, and Sen [24, 25] (also Jaiswal, Kumar, and Sen [23]). In this approach, one first takes a large enough sample that can be used to pin down the location of the largest cluster center. Then, one removes enough points from the vicinity of this center, so that the next largest cluster becomes dominant, and hence a subsequent sample contains sufficiently many points from the second cluster, and so on.

However, our scenario is more intricate and challenging for several reasons due to the peculiar nature of the objective. Nevertheless, in principle, one can classify each cluster as either being more *1-center-like*, or more *1-median like* (see Figure 3 for an illustration). In a *1-center-like* cluster, a large fraction of points lie within a ball of radius  $\mathcal{O}(r/\varepsilon)$ . On the other hand, in a *1-median-like* cluster, a vast majority of points lie outside the  $\mathcal{O}(r/\varepsilon)$ -radius ball. Note that any such point loses very little due to the “ $-r$ ” term in the clustering cost, i.e., its  $\text{dist}_r$  and  $\text{dist}$  values are approximately equal. Hypothetically, if we knew the partition of the input points into  $k$  clusters, then we could use this classification to handle each type of cluster separately – an almost-optimal center of a *1-center-like* cluster can be found using a grid, whereas one can use an approximation for 1-median (as a black box) to handle a *1-median-like* cluster. However, the actual clusters are obviously unknown to the algorithm. Hence, the algorithm has to carefully navigate between the two types of clusters based on the random sample obtained, and must simultaneously handle both scenarios using branching (i.e., recursion). The analysis of the algorithm is also much more involved due to the various cases in which the distinction between two types of clusters is murkier. Nevertheless, we are able to show that the algorithm returns a  $(1 + \varepsilon, 1 + \varepsilon)$ -approximation in time  $2^{(kd/\varepsilon)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$  with good probability. Note that we incur an exponential dependence on the dimension  $d$  due to “grid-arguments” used to handle *1-center-like* clusters, unlike the approach of [24]. However, such dependence seems unavoidable using our approach.





■ **Figure 3** Example of two different types of clusters. In each figure, we show the cluster center in red, a ball of radius  $r$  around the center in green, and a larger ball of radius  $\mathcal{O}(r/\epsilon)$  in cyan with a dashed outline. Left: A *1-center-like* cluster. Note that a large chunk of points lies within the radius  $\mathcal{O}(r/\epsilon)$  ball around the center. Right: A *1-median-like* cluster. Note that most of the points lie outside the  $\mathcal{O}(r/\epsilon)$  radius ball around  $c$ , and for any such point, e.g.,  $p$  that is outside the  $\mathcal{O}(r/\epsilon)$  radius ball,  $\text{dist}_r(p, c) \approx \text{dist}(p, c)$ .

## 1.2 Related Problems

**Euclidean Clustering.** An extensive body of literature exists on approximation algorithms for  $k$ -CENTER and  $k$ -MEDIAN in the Euclidean space. For  $k$ -MEDIAN in  $\mathbb{R}^d$ , Polynomial-Time Approximation Schemes (PTASes) with a running time of  $n^{f(\epsilon, d)}$  have been developed, leveraging local search techniques [18, 12]. Additionally, various Fixed-Parameter Tractable Approximation Schemes (FPT-AS) with a running time of  $f(k, d, \epsilon) \cdot n^{\mathcal{O}(1)}$  are known for this problem [11, 13, 25, 23]. The dependence on dimension  $d$  can be eliminated through dimensionality reduction techniques [26, 9].

For  $k$ -CENTER, an FPT-AS was introduced by Agarwal and Procopiuc in [4], with a runtime of  $\mathcal{O}(n \log k) + (k/\epsilon)^{\mathcal{O}(dk^{1-1/d})}$  in  $\mathbb{R}^d$ . Subsequent work by Badoiu, Har-Peled, and Indyk [5] improved the running time to  $2^{\mathcal{O}(k \log k)}/\epsilon^2$ .

In [30], Tamir introduces a common generalization of the two clustering problems, namely,  $\ell$ -CENTRUM. In this problem, one ignores  $\ell$  closest points from the cost. Notably,  $k$ -MEDIAN ignores 0 points, and  $k$ -CENTER ignores all but one point. While this problem is related to HYBRID  $k$ -CLUSTERING, their objectives differ. ORDERED  $k$ -MEDIAN, a further generalization of  $\ell$ -CENTRUM, also does not align with our objective. Approximation algorithms for this problem and some variants were developed in [2, 1, 6, 7].

**$k$ -center clustering with outliers.** In  $k$ -CENTER clustering, we are given sets  $P$  (clients) and  $\mathbb{F}$  (facilities) of points. Given an integer  $k$ , the task is to identify  $k$  centers  $F \subseteq \mathbb{F}$  minimizing the maximum distance of any point in  $P$  from its closest center. A popular variant of  $k$ -center is a formulation that considers outliers. For a selected parameter  $x$ , up to  $x$  points are allowed not to be allocated to any center. A plethora of approximation algorithms for this problem, and the related problems of covering points by disks and minimum enclosing balls

with outliers, exist in the literature [5, 14, 8, 19, 15, 16, 21, 28]. HYBRID  $k$ -CLUSTERING could be seen as a variant of  $k$ -CENTER with outliers, where we focus on the sum of distances to outliers rather than their numbers.

**Shape fitting.** A natural problem arising in machine learning, statistics, data-mining, and many other fields is to fit a shape  $\gamma$  to a set of points  $P$  in  $\mathbb{R}^d$ . Har-Peled in [20] introduces the following formalization of this problem. For a family of shapes  $\mathcal{F}$  (points, lines, hyperplanes, spheres, etc.) we seek for a shape  $\gamma \in \mathcal{F}$  with the best fit to  $P$ . The typical criteria for measuring how well a shape  $\gamma$  fits a set of points  $P$  could be the maximum distance between a point of  $P$  and its nearest point on  $\gamma$  ( $L_\infty$ -fitting), sum of the distances from  $P$  to  $\gamma$  ( $L_1$ -fitting) or the sum of the squares of the distances ( $L_2$ -fitting). In this setting, HYBRID  $k$ -CLUSTERING is the problem of  $L_1$ -fitting to a shape from  $\mathcal{F}$ , where  $\mathcal{F}$  is the family of shapes defined by unions (not necessarily disjoint) of  $k$  balls in  $\mathbb{R}^d$ . Some relevant work in this direction includes [3, 20, 22, 31].

## 2 Bicriteria FPT Approximation Scheme in Euclidean Spaces

We first set up some notation and define an important subroutine. For  $Y \subset \mathbb{R}^d$ , and  $y \in Y$ , let  $\text{cl}(y, Y) \subseteq P$  denote the subset of points of  $P$ , whose closest point in  $Y$  is  $y$ . Ties are broken arbitrarily. Note that  $\{\text{cl}(y, Y) : y \in Y\}$  forms a partition of  $P$ .

Let  $p \in \mathbb{R}^d$  be a point and  $\lambda \geq 0$ , let  $B(p, \lambda) = \{q \in \mathbb{R}^d : d(p, q) \leq \lambda\}$  denote the ball of radius  $\lambda$  centered at  $p$ . For  $0 \leq \tau \leq \lambda$ , let  $\text{Grid}(p, \lambda, \tau)$  be the outcome of the following procedure: we place a grid of sidelength  $\tau/\sqrt{d}$  (of arbitrary offset). From each grid cell  $L$  that (partially) intersects with  $B(p, \lambda)$  (i.e.,  $L$  contains a point  $q$  with  $d(p, q) \leq \lambda$ ), we pick an arbitrary point from  $L$  and add it to the set  $\text{Grid}(p, \lambda, \tau)$ . Note that  $\text{Grid}(p, \lambda, \tau)$  can be computed in time proportional to the size of the output. We have the following observations that follow from simple geometric arguments.

### ► Observation 3.

1.  $|\text{Grid}(p, \lambda, \tau)| \leq \mathcal{O}((\sqrt{d}\lambda/\tau)^d)$ , where  $d$  is the dimension.
2. For any  $q \in B(p, \lambda)$ , there exists some  $q' \in \text{Grid}(p, \lambda, \tau)$  such that  $d(q, q') \leq \tau$ .

## 2.1 Preprocessing

Suppose we know an estimate of  $\text{OPT}_r$  up to a constant factor – this can be done by an exponential search or by first finding a bicriteria (constant) approximation. For simplicity of exposition, we assume that we know  $\text{OPT}_r$  exactly.

**Step 1.** Obtaining  $\text{OPT}_r \geq r \geq \frac{\varepsilon \text{OPT}_r}{2n}$ .

First, in the following lemma, we handle  $k$ -center-like instances, which we can handle using “grid arguments”. If this is not applicable, we obtain that  $r \leq \text{OPT}_r$ .<sup>1</sup>

► **Lemma 4 (♠).** *If  $r > \text{OPT}_r$ , then in time  $(\frac{d}{\varepsilon})^{\mathcal{O}(dk)} \cdot n^{\mathcal{O}(1)}$  one can find a set  $F \subset \mathbb{R}^d$  of size  $k$ , such that,  $\text{cost}_{(1+\varepsilon)r}(P, F) \leq (1 + \varepsilon)\text{OPT}_r$ .*

<sup>1</sup> Due to space constraints, some proofs are omitted in this extended abstract. They can be found in the full version of the paper [17]. The statements with missing proofs are marked by ♠.

In the following lemma, we handle *k*-median-like instances, where  $r$  is very small compared to  $\text{OPT}_r$ . We directly reduce such instances to *k*-MEDIAN (where  $r = 0$ ). If this is not applicable, then we obtain that  $r$  is not “too small” compared to  $\text{OPT}_r$ .

► **Lemma 5** (♠). *Let  $0 < \varepsilon < 1$ . Suppose for an instance  $\mathcal{I}$ ,  $\text{OPT}_r \geq \frac{2nr}{\varepsilon}$ . Then,  $\text{OPT}_0 \leq (1 + \varepsilon/2) \cdot \text{OPT}_r \leq (1 + \varepsilon/2) \cdot \text{OPT}_0$ . Furthermore, if  $F \subset \mathbb{R}^d$  satisfies that  $\text{cost}_0(P, F) \leq (1 + \varepsilon/3) \cdot \text{OPT}_0$ . Then,  $\text{cost}_r(P, F) \leq (1 + \varepsilon) \cdot \text{OPT}_r$ . Such a set  $F$  can be found in time  $2^{(k/\varepsilon)^{\mathcal{O}(1)}} \cdot nd$ .*

For a given input  $P$ , we try the procedures from Lemma 4 and 5 and keep them as candidate solutions. However, if  $P$  does not satisfy the conditions required to apply these lemmas, then we must have that  $\frac{\varepsilon \text{OPT}_r}{2n} \leq r \leq \text{OPT}_r$ . In this case, we use the next step before proceeding to the main algorithm.

**Step 2.** Bounding the aspect ratio.

In this step, we suitably discretize the distances in order to bound the aspect ratio of the metric (i.e., the maximum ratio of inter-point distances) by  $\mathcal{O}(\frac{n^2}{\varepsilon})$ . This procedure preserves the cost of an optimal solution up to a factor of  $1 + \varepsilon$ .

► **Lemma 6** (♠). *Let  $P$  be a set of points satisfying  $\frac{\varepsilon \text{OPT}_r}{2n} \leq r \leq \text{OPT}_r$ . Then, in polynomial time we can obtain another (multi)set of points  $P'$  such that, for any solution  $F \subset \mathbb{R}^d$ ,  $\text{cost}_r(P', F) \in (1 \pm \varepsilon) \cdot \text{cost}_r(P, F)$ , and  $\frac{\max_{p,q \in P'} \text{dist}(p,q)}{\min_{p,q \in P': \text{dist}(p,q) \neq 0} \text{dist}(p,q)} \leq \frac{4n^2}{\varepsilon}$ .*

Bounding the aspect ratio by  $\mathcal{O}(\frac{n^2}{\varepsilon})$  means that an exponential search over distances has at most  $\log_2 \left( \frac{n^2}{\varepsilon} \right) = \mathcal{O} \left( \frac{\log(n)}{\varepsilon} \right)$  levels, which will be useful in our main algorithm. By slightly abusing the notation, we continue to use  $P$  for referring to the discretized (multi)set  $P'$  returned by Lemma 6. If there are any co-located points in  $P$ , we will treat them as separate points, and hence use set terminology instead of multiset terminology.

After the two preprocessing steps, we now proceed to the description of the main algorithm.

## 2.2 Main Algorithm

Our goal is to prove Theorem 2, that is, to design a randomized bicriteria FPT approximation for HYBRID *k*-CLUSTERING. We define some parameters. Let  $\delta := \frac{\varepsilon}{10k} < \frac{1}{2}$ ,  $\delta' := \frac{\delta}{3}$  and  $r' := (1 + \delta')r$ .

Algorithm 1 is a recursive algorithm, and is called `HybridClustering`. It takes three parameters  $F', k$ , and  $m$ .  $F' \subset \mathbb{R}^d$  is a subset of centers added to the solution so far and has size  $k - m$ . Further,  $k$  is the total size of the solution, and  $m$  is an upper bound on the remaining solution (since we have already added  $k - m$  centers). At a high (and imprecise) level, the goal of each recursive step is to find an *approximate replacement* for each center in an unknown optimal solution.

In line 2, we check whether  $m = 0$ , i.e., whether we have used our budget of  $k$  centers, and if so, we return the same set  $F'$  of centers built through the recursive process. Otherwise (line 4 onward), we assume that  $m > 0$ , i.e., we are yet to add a set of centers. Throughout this process (line 4 to 13, we will build a set  $R$  consisting of candidate centers, at least one of which will be an approximate replacement of an *unseen* center (i.e., one whose approximate replacement has not already been found) from an optimal solution. Finally, in line 15, we will make a recursive call by adding each candidate to the current solution  $F'$ . Now we discuss how we build the set  $R$ .

---

**Algorithm 1** HybridClustering( $F', k, m$ ).
 

---

$F' \subseteq \mathbb{R}^d$  is a subset of centers of size at most  $k - m$  added to the solution so far  
 $\beta = \frac{1}{\delta^{c'}} r$  as required in Proposition 13 and  $\beta' := \beta \cdot \frac{150k}{\delta^3}$ .

- 1: **if**  $m = 0$  **then**
- 2:     **return**  $F'$
- 3: **end if**
- 4:  $R \leftarrow \bigcup_{c' \in F'} \text{Grid}(c', 16r, \delta r)$
- 5: **for** each  $q$  of the form  $2^j$  in the range  $[8r, \text{dist}_{\max}]$  **do**
- 6:      $P_q := P \setminus (\bigcup_{c' \in F'} B(c', q))$
- 7:     Let  $S_q$  be a sample of size  $\beta'$  chosen uniformly at random from  $P_q$
- 8:      $R \leftarrow R \cup \bigcup_{p \in S_q} \text{Grid}(p, \frac{8r}{\delta}, \delta r)$
- 9:     **for** each  $S \subseteq S_q$  of size  $\beta$  **do**
- 10:          $c' \leftarrow \text{ApproxSolutionOnSample}(S, \delta/8)$       $\triangleright$  Algorithm from Proposition 13
- 11:          $R \leftarrow R \cup \{c'\}$
- 12:     **end for**
- 13: **end for**
- 14: **for** each  $c \in R \setminus F'$  **do**
- 15:     Call HybridClustering( $F' \cup \{c\}, k, m - 1$ )
- 16: **end for**
- 17: Call HybridClustering( $F', k, m - 1$ )
- 18: **return** solution  $\tilde{F}$  minimizing  $\text{cost}_{r'}(P, \tilde{F})$  over recursive calls made in lines 15 and 17

---

First, in line 4, for each center  $c' \in F'$  added so far, we add a set of “nearby” centers by placing a grid. This handles the case when an unseen optimal center is close to one of the already chosen centers in  $F'$ . Next, in the outer **for** loop (line 5 to 13), we handle the case when all new optimal centers are relatively far from the already chosen centers. In this **for** loop, we iterate over a range of values for the parameter  $q$  via exponential search. Parameter  $q$  tries to approximate half of the minimum distance between the already chosen and new optimal centers. Thus, for the “correct” value of  $q$ , the set of points  $C_q$  lying “far” from the centers of  $F'$  (line 6), leaves all of the  $m$  unseen optimal clusters untouched. At this point, we aim to use a sample of faraway size (chosen in line 7), to find an approximate replacement for one of these  $m$  unseen centers. We do this by using the sample in two different ways, to handle two different situations. First, if our sample happens to contain a point “nearby” an unseen center, say  $c^*$ , then the points chosen from the fine grid in line 8 will find such an approximate replacement for  $c^*$ . Otherwise, the idea is that, if we have removed a significant fraction of points from the “seen” clusters in line 6, by virtue of being close to  $F'$ , then the sample contains sufficiently many (i.e., at least  $\beta$ ) points from the largest unseen cluster, say  $C^*$ , with reasonable probability, and these points can be used to find an approximate replacement of the cluster center (using Proposition 13). However, *a priori* we do not know which subset of the sample comes from  $C^*$ . Therefore, we iterate over all subsets of size  $\beta$  in the inner **for** loop (lines 9 to 12) to find such a subset of size  $\beta$  that comes entirely from  $C^*$  and use a known subroutine, called **ApproxSolutionOnSample**, to find an approximate replacement. Finally, in 15, we make a recursive call by adding each center from  $R \setminus F'$ , and in line 17, we make a recursive call by not adding any new center (to handle a particular case). In line 18, we return the minimum-cost solution found over all recursive calls. This completes the description of the algorithm.

## 2.3 Analysis

The crux of the analysis is to establish that Algorithm 1 satisfies the following invariant.

### Invariant

Let  $0 \leq m \leq k$  and  $0 < \alpha < 1$  be a constant. Suppose for the given  $F'$  of size at most  $k - m$ , there exists some  $F = F' \uplus F_o \subset \mathbb{R}^d$ , such that

1.  $|F_o| \leq m$ , and
- 2.

$$\sum_{c \in F'} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) \leq (1 + \delta)^{k-m} \cdot \text{OPT}_r \quad (2)$$

Then, with probability at least  $\alpha^m$ , the algorithm returns a solution  $\tilde{F} \subset \mathbb{R}^d$ , such that

1.  $|\tilde{F}| \leq k$ ,
2.  $F' \subseteq \tilde{F}$ , and
- 3.

$$\sum_{c \in \tilde{F}} \text{cost}_{r'}(\text{cl}(c, F), c) \leq (1 + \delta)^k \cdot \text{OPT}_r. \quad (3)$$

### Proof of Correctness

The proof is by induction on  $m$ . For the base case, consider  $m = 0$ . In the base case (Algorithm 1), we return the same  $F' = F$  with probability one. In this case, the invariant tells us that  $\text{cost}_{r'}(P, F) \leq (1 + \delta)^k \cdot \text{OPT}_r$ , which is what we need to prove. Now we assume that the claim is true for some  $m - 1 \geq 0$  and we prove it for  $m$  by considering different cases.

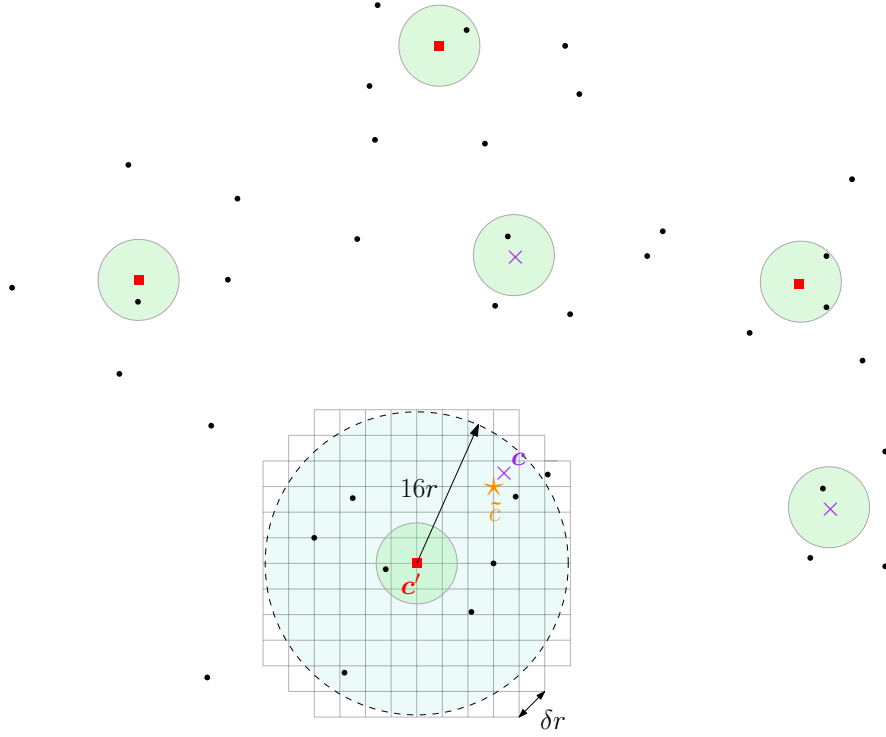
**Easy case:  $F = F'$ .** This is a much simpler case since we have already found the desired set. In this case, any solution  $\tilde{F}$  returned by a recursive call always contains  $F = F'$  as a subset. Then, in this case, we have that:

$$\text{cost}_{r'}(P, \tilde{F}) \leq \text{cost}_{r'}(P, F) \leq (1 + \delta)^{k-m} \cdot \text{OPT}_r \leq (1 + \delta)^k \cdot \text{OPT}_r$$

Here, the first inequality follows from the assumption that  $\tilde{F} \supseteq F = F'$ , and the second inequality follows from (2) of the invariant. Note that we do not need to rely on the induction here.

**Main case:  $F' \subsetneq F$ .** This is the case where we are yet to discover some subset (namely,  $F \setminus F'$ ) of centers. We will analyze this case by considering different scenarios based on the inter-center distances, as well as their relative sizes.

First, since  $F' \subsetneq F$ , there exists some  $c \in F \setminus F'$ . Now, let  $c \in F \setminus F'$  and  $c' \in F'$  be the pair of centers with the smallest distance, i.e.,  $(c, c')$  is a pair realizing  $\min_{c_1 \in F \setminus F', c_2 \in F'} \text{dist}(c_1, c_2)$ . Now we consider different cases depending on  $\text{dist}(c, c')$ , namely the closest distance between an already chosen center  $c' \in F'$ , and an “unseen center”  $c \in F \setminus F'$ .



■ **Figure 4** Illustration for Case 1. Centers in  $F'$  are shown as red squares and unseen centers of  $F \setminus F'$  are shown as purple crosses.  $c$  is the closest center to  $F'$  and  $\text{dist}(c, c') \leq 16r$ . Then, a nearby center  $\tilde{c}'$  can be found using a  $\delta r$  grid.

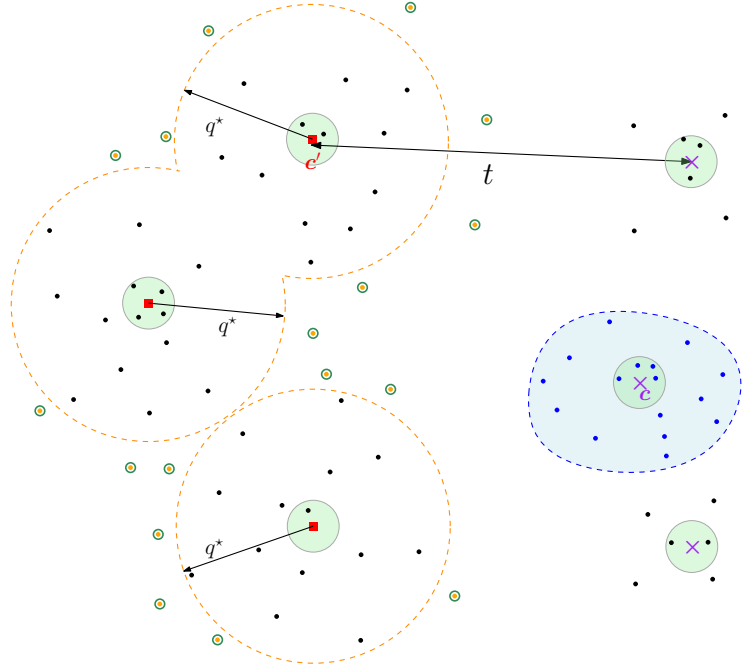
**Case 1. Nearby center:  $\text{dist}(c, c') \leq 16r$ .** In this case, via Observation 3, we conclude that there exists some  $\tilde{c} \in \text{Grid}(c', 16r, \delta r)$  with  $\text{dist}(\tilde{c}, c) \leq \delta r$ . Let  $\tilde{F} = F' \cup \{\tilde{c}\}$ . Then, the proof follows from the following claim (see Figure 4).

▷ **Claim 7.** Let  $c_1 \in F_o$  and let  $\tilde{c}_1 \in \mathbb{R}^d$  be such that  $\text{dist}(c_1, \tilde{c}_1) \leq \delta r$ . Then, with probability at least  $\alpha^{m-1}$ ,  $\text{HybridClustering}(F \cup \tilde{c}_1, k, m - 1)$  returns a solution  $\tilde{F}$  that satisfies the required properties.

*Proof.* Consider  $c_1, \tilde{c}_1$  as defined in the statement. Let  $A = \text{cl}(c_1, F)$ . For any point  $p \in A$ ,  $\text{dist}(p, \tilde{c}_1) \leq \text{dist}(p, c_1) + \text{dist}(c_1, \tilde{c}_1) \leq \text{dist}(p, c_1) + \delta r$ . This implies that,  $\text{dist}_{r'}(p, \tilde{c}_1) \leq \text{dist}_{r'}(p, c_1)$ . Define  $F_{\text{new}} := F'_{\text{new}} \uplus F'_o$ , where  $F'_{\text{new}} := F' \cup \{\tilde{c}_1\}$  and  $F'_o := F_o \setminus \{c_1\}$ . First, we show the following inequality.

$$\begin{aligned} \sum_{c \in F'_{\text{new}}} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F'_o} \text{cost}_r(\text{cl}(c, F_{\text{new}}), c) \\ \leq \sum_{c \in F'} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) \end{aligned} \quad (4)$$

We construct an assignment of clients to the centers in  $F_{\text{new}}$ , where we may not assign a client to its closest center. To construct this assignment, we consider different cases. For  $c \in F' \cup F_o \setminus \{c_1\}$ , we assign all points  $p \in \text{cl}(c, F)$  to  $c$ . The contribution of all such points is the same as the right-hand side of (4). Finally, we assign all points in  $\text{cl}(c_1, F)$  to  $\tilde{c}_1$ . By the choice of  $\tilde{c}_1$ ,  $\text{cost}_{r'}(\text{cl}(c_1, F), \tilde{c}_1) \leq \text{cost}_r(\text{cl}(c_1, F), c_1)$ , which is the contribution of such points on the right-hand side. Since the cost on the left-hand side is no larger than the cost of the assignment thus constructed, it shows (4).



■ **Figure 5** Illustration for Case 2. Centers of  $F'$  are shown as red squares and unseen centers of  $F \setminus F'$  are shown as purple crosses. Balls of radius  $q^*$  around  $F'$  are shown in dashed orange.  $P'$  are the points lying outside these balls. Among the points of  $P'$ ,  $D$  is the set of points belonging to clusters around  $F'$ , and shown as green-orange filled dots. Finally, the cluster around  $c$  is the largest unseen cluster (marked in dashed blue shape),  $L$ . We analyze different cases depending on the relative sizes of  $L$  and  $D$ .

Note that the right-hand side of (4) is at most  $(1 + \delta)^{k-m} \cdot \text{OPT}_r$ , due to the invariant, and hence  $F_{\text{new}}$  satisfies the properties required to apply the inductive hypothesis for  $m - 1$ . This implies that with probability at least  $\alpha^{m-1}$  the recursive call  $\text{Recursive}(F' \cup \{\tilde{c}_1\}, k, m - 1)$  returns a solution  $\tilde{F}$  satisfying  $\text{cost}_{r'}(P, \tilde{F}) \leq (1 + \delta)^k \cdot \text{OPT}_r$ .  $\triangleleft$

**Case 2. Faraway center:**  $16r < \text{dist}(c, c') \leq \text{dist}_{\max}$ . Let  $t = \text{dist}(c, c')$  and  $q^*$  be the largest power of 2 that is at most  $t/2$ . Consider  $P_{q^*} = P \setminus (\bigcup_{c_1 \in F'} B(c_1, q^*))$ . Let  $c^* \in F \setminus F'$  denote the center of the maximum-size cluster, i.e.,  $c^* = \arg \max_{c_1 \in F \setminus F'} |\text{cl}(c_1, F)|$ , and  $L := \text{cl}(c^*, F)$  denote the largest cluster. Finally, let  $D := \bigcup_{c_{\text{old}} \in F'} \text{cl}(c_{\text{old}}, F) \cap P_{q^*}$  denote the set of clients that are distant from the respective centers in  $F'$ . Let us summarize some consequences of these definitions in the following observation (its proof is essentially discussed above). Also see Figure 5.

► **Observation 8.**

1.  $P_{q^*} = D \uplus \biguplus_{c_1 \in F \setminus F'} \text{cl}(c_1, F) \cap P_{q^*}$ .
2. In particular,  $\text{cl}(c, F), \text{cl}(c^*, F) \subseteq P_{q^*}$ .

We consider different sub-cases based on the relative sizes of  $L$  and  $D$ .

**Case 2.1. New cluster is tiny:**  $|L| \leq \delta^2/4 \cdot |D|$ . Let  $N := \text{cl}(c, F)$ . Note that the definition of  $c^*$ , combined with the case assumption, implies that  $|N| \leq |L| \leq \frac{\delta^2}{4} \cdot |D|$ . We summarize a few technical consequences of these definitions in the following claim.

## 4:12 Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center

▷ Claim 9.

$$\text{cost}_r(N, c') \leq \delta \cdot \text{cost}_r(D, F) + (1 + \delta) \cdot \text{cost}_r(N, c) \quad (5)$$

Proof. Note that for each  $p \in D$ ,  $\text{dist}(p, F) \geq \frac{t}{4}$ . Thus,  $\text{dist}_r(p, c') \geq \frac{t}{4} - r \geq \frac{3t}{16}$ , where the last inequality follows from the case assumption, namely  $t > 16r$ . Thus, each point  $p \in D$  contributes at least  $\frac{3t}{16}$  to  $\text{cost}_{r'}(P, F)$ , and their total contribution to  $\text{cost}_r(P, F)$  is

$$\text{cost}_r(D, F) := \sum_{p \in D} \text{dist}_r(p, c') \geq |D| \cdot \frac{3t}{16} \quad (6)$$

Now we upper bound the cost of assigning points of  $N$  to  $c'$ . To this end, we partition  $N = N_{\text{near}} \uplus N_{\text{far}}$ , where  $N_{\text{near}} := \{p \in A : \text{dist}(p, c) \leq 2r/\delta\}$  and  $N_{\text{far}} := \{p \in A : \text{dist}(p, c) > 2r/\delta\}$ . Note that, for each  $p \in N_{\text{far}}$ ,  $\text{dist}_r(p, c) = \text{dist}(p, c) - r \geq \text{dist}(p, c) - \frac{\delta}{2} \cdot \text{dist}(p, c)$ , which implies that, for each  $p \in N_{\text{far}}$ ,

$$\text{dist}(p, c) \leq \left( \frac{1}{1 - \frac{\delta}{2}} \right) \cdot \text{dist}_r(p, c) \leq (1 + \delta) \cdot \text{dist}_r(p, c) \quad (7)$$

Now consider,

$$\begin{aligned} \text{cost}_r(N, c') &\leq \sum_{p \in N} \text{dist}(p, c') && \text{(Since } \text{dist}_r(\cdot, \cdot) \leq \text{dist}(\cdot, \cdot)\text{)} \\ &\leq \sum_{p \in N} \text{dist}(p, c) + \text{dist}(c, c') && \text{(Triangle inequality)} \\ &= |N| \cdot \text{dist}(c, c') + \sum_{p \in N_{\text{near}}} \text{dist}(p, c) + \sum_{p \in N_{\text{far}}} \text{dist}(p, c) \\ &\leq \frac{\delta^2}{8} \cdot |D| \cdot t + \sum_{p \in N_{\text{near}}} \frac{4r}{\delta} + \sum_{p \in N_{\text{far}}} (1 + \delta) \cdot \text{dist}_r(p, c) \\ &&& \text{(From case assumption and (7))} \\ &= \frac{\delta^2}{4} \cdot |D| \cdot t + \frac{4r}{\delta} \cdot |N_{\text{near}}| + (1 + \delta) \cdot \text{cost}_r(N, c) \\ &\leq \frac{\delta^2}{4} \cdot |D| \cdot t + \delta \cdot |D| \cdot \frac{t}{16} + (1 + \delta) \cdot \text{cost}_r(N, c) \\ &&& (|N_{\text{near}}| \leq |N| \leq \frac{\delta^2}{4} \text{ and } t > 16r) \\ &\leq |D| \cdot \frac{3t}{16} \cdot \delta \cdot \left( \frac{4\delta}{3} + \frac{1}{3} \right) + (1 + \delta) \cdot \text{cost}_r(N, c) \\ &\leq \delta \cdot \text{cost}_r(D, F) + (1 + \delta) \cdot \text{cost}_r(N, c) \end{aligned} \quad (8)$$

Where the last inequality follows from (6) and  $\delta < 1/2$ . ◁

Thus, consider the solution  $F \setminus \{c\}$ . To upper bound  $\text{cost}_r(P, F \setminus \{c\})$ , we assign all points in  $N = \text{cl}(c, F)$  to  $c'$ . The cost of this solution can be upper bounded as follows

$$\begin{aligned} \text{cost}_r(P, F \setminus \{c\}) &\leq \text{cost}_r(P, F) - \text{cost}_r(N, c) + \text{cost}_r(N, c') \\ &\leq \text{cost}_r(P, F) + \delta \cdot \text{cost}_r(D, F) + \delta \cdot \text{cost}_r(N, c) && \text{(From (5))} \\ &\leq \text{cost}_r(P, F) + \delta \cdot \text{cost}_r(P, F) && \text{(Since } D \uplus N \subseteq P\text{)} \\ &\leq (1 + \delta)^{k-m+1} \cdot \text{OPT}_r \end{aligned} \quad (9)$$

Where the last inequality follows from the invariant. Further, observe that  $c \in F \setminus F'$ , which implies that  $|(F \setminus \{c\}) \setminus F'| \leq m - 1$ . This, combined with (9), shows that the solution  $F \setminus \{c\} = F' \uplus (F_o \setminus \{c\})$  satisfies the conditions of the invariant for  $m - 1$ . Then, by using inductive hypothesis,  $\text{HybridClustering}(F, k, m - 1)$ , with probability at least  $\alpha^{m-1} \geq \alpha^m$ , returns a solution  $\tilde{F}$  such that (a)  $|\tilde{F}| \leq k$ , (b)  $F \subseteq \tilde{F}$ , and (c)  $\text{cost}_{(1+\delta)r'}(F, \tilde{F}) \leq (1 + \delta)^k \cdot \text{OPT}_r$ , completing the induction.



**Case 2.2. New cluster is large enough:**  $|L| > \delta^2/4 \cdot |D|$ . That is, the largest “untouched” cluster is at least an  $\delta$  fraction of the remaining points. Since  $L = \text{cl}(c^*, F)$  is the largest “untouched” cluster,  $|L| \geq |\text{cl}(c_1, F)|$  for all  $c_1 \in F \setminus F'$ . Then, by Observation 8, we have that,

$$|P_{q^*}| = |D| + \sum_{c_1 \in F \setminus F'} |\text{cl}(c_1, F)| \leq |D| + |F \setminus F'| \cdot |L| \leq \frac{4}{\delta^2} \cdot |L| + k \cdot |L| \leq \frac{5k}{\delta^2} \cdot |L|$$

In other words,  $|L| \geq \frac{\delta^2}{5k} \cdot |P_{q^*}|$ .

In the next claim, we summarize some properties of the sample  $S$  chosen in line 7 of the algorithm, in the current case, i.e., when  $|L| \geq \frac{\delta^2}{5k} |C_{q^*}|$ .

▷ **Claim 10.** Consider the iteration of the **for** loop of Algorithm 1, when  $q = q^*$ , and the corresponding sample  $S_q$  obtained in Algorithm 1. The following statements hold.

1. With probability at least  $1/2$ ,  $S_q$  contains at least  $\beta$  points of  $L$ .
2.  $S_q \cap L$  has the same distribution as selecting  $|S_q \cap L|$  points uniformly at random from  $L$ .
3. Let  $L' \subseteq L$  be an arbitrary subset of size at least  $\frac{\delta}{10}|L|$ . Then, with probability at least  $1/2$ ,  $S_q$  contains at least 1 point from  $L$ , i.e.,  $S_q \cap L' \neq \emptyset$ .

*Proof.* Recall that  $\beta' = \beta \cdot \frac{150k}{\delta^3}$  and  $|L| \geq \frac{\delta^2}{5k} |C_{q^*}|$ . So, the first item follows, say, via Markov’s inequality<sup>2</sup>. The second item is an easy consequence of conditional distributions. The proof of the third item is analogous to the first item, combined with the bound on  $|L'|$ . ◁

Now we condition on the event that  $|S_{q^*} \cap L| \geq \beta$ , which, by Claim 10 happens with probability at least  $\frac{1}{2}$ . Then, let  $S' \subseteq S \cap L$  be such a subset of size  $\beta$ . Let  $L = L_{\text{near}} \uplus L_{\text{far}}$ , where  $L_{\text{near}} = \{p \in L : \text{dist}(p, c^*) \leq \frac{8r}{\delta}\}$  and  $L_{\text{far}} = \{p \in L : \text{dist}(p, c^*) > \frac{8r}{\delta}\}$ . We consider different cases depending on the relative sizes of  $L_{\text{near}}$  and  $L_{\text{far}}$ . In the first case below (2.2.1), when  $|L_{\text{near}}|$  is not very tiny compared to  $|L_{\text{far}}|$ , we show that our sample contains at least one point from  $L_{\text{near}}$  with good probability, and hence an  $\varepsilon r$  grid around that point will contain an approximate center. In the complementary case (2.2.2),  $|L_{\text{near}}|$  is very tiny compared to  $|L_{\text{far}}|$ , and in this case, we argue that, instead of finding an approximate HYBRID 1-MEDIAN, we can focus on finding an approximate 1-MEDIAN, which can be found using the sample. Now we formally analyze each of these cases.

**Case 2.2.1.**  $|L_{\text{near}}| > \frac{\delta}{8} \cdot |L_{\text{far}}|$ . In this case, letting  $L' \leftarrow L_{\text{near}}$  in Claim 10, we infer that with at least probability  $1/2$ ,  $S' \cap L_{\text{near}} \neq \emptyset$ . We condition on this event. Then, since  $\text{dist}(p, c^*) \leq \frac{8r}{\delta}$ , it follows that there exists a  $\tilde{c}^* \in \text{Grid}(p, \frac{8r}{\delta}, \delta r)$ , such that  $\text{dist}(\tilde{c}^*, c^*) \leq \delta r$ . Since we branch on each point in  $\bigcup_{p' \in S} \text{Grid}(p', \frac{8r}{\delta}, \delta r)$ , we will branch on  $\tilde{c}^*$  in particular. Then, by Claim 7,  $\text{HybridClustering}(F' \cup \{\tilde{c}^*\}, k, m - 1)$  returns a solution  $\tilde{F}$ , with probability at least  $1/2 \cdot \alpha^{m-1} \geq \alpha^m$ .

**Case 2.2.2.**  $|L_{\text{near}}| \leq \frac{\delta}{8} \cdot |L_{\text{far}}|$ . We prove two claims, namely Claim 11, and Claim 12. The latter essentially reduces the problem to finding an approximate solution to 1-median on  $L$ . Intuitively speaking, this follows from the following two reasons: (1) As we show in (10), a similar statement holds for the points in  $L_{\text{far}}$ . This essentially follows from the fact that, since each point of  $L_{\text{far}}$  has distance at least  $\frac{8r}{\delta}$  to  $c$ , the subtraction of  $r$  from their distances has little effect on the cost, and (2) Due to the case assumption, the points of  $L_{\text{far}}$  vastly outnumber the points of  $L_{\text{near}}$ . Hence, the preceding claim also translates to the points of  $L = L_{\text{far}} \cup L_{\text{near}}$ , at a further small approximation error.

<sup>2</sup> In fact, a closer inspection reveals that the probability is much closer to 1, but “at least  $1/2$ ” suffices for our purpose.

#### 4:14 Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center

▷ Claim 11.  $\sum_{p \in L} \text{dist}_r(p, c^*) \leq \sum_{p \in L} \text{dist}(p, c^*) \leq (1 + \frac{3\delta}{4}) \cdot \sum_{p \in L} \text{dist}_r(p, c^*)$ .

Proof. First, consider,

$$\sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) = \sum_{p \in L_{\text{far}}} \text{dist}(p, c^*) - r \geq \sum_{p \in L_{\text{far}}} \text{dist}(p, c^*) - \frac{\delta}{8} \cdot \text{dist}(p, c^*)$$

(Since  $\text{dist}(p, c^*) \geq \frac{8r}{\delta} > r$ )

Then, the inequality between the first and the last term can be rewritten as,

$$\sum_{p \in L_{\text{far}}} \text{dist}(p, c^*) \leq \frac{1}{1 - \delta/8} \cdot \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) \leq (1 + \frac{\delta}{4}) \cdot \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) \quad (10)$$

The following inequality will be used later to show that the contribution of points of  $L_{\text{near}}$  is negligible to the overall cost.

$$\begin{aligned} \sum_{p \in L_{\text{far}}} \text{dist}(p, c^*) &\geq \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) \geq \sum_{p \in L_{\text{far}}} (1 - \frac{\delta}{8}) \cdot \text{dist}(p, c^*) && \text{(From (10))} \\ &\geq (1 - \frac{\delta}{8}) \cdot \frac{8r}{\delta} \cdot |L_{\text{far}}| && \text{(Definition of } L_{\text{far}}) \\ &\geq \frac{1}{2} \cdot \frac{8r}{\delta} \cdot \frac{8}{\delta} \cdot |L_{\text{near}}| && \text{(Case assumption: } |L_{\text{far}}| \geq \frac{8}{\delta} \cdot |L_{\text{near}}|) \\ &= \frac{4}{\delta} \cdot \frac{8r}{\delta} \cdot |L_{\text{near}}| \\ &\geq \frac{4}{\delta} \cdot \sum_{p \in L_{\text{near}}} \text{dist}_r(p, c^*) && (11) \end{aligned}$$

Where the last inequality follows from the definition of  $L_{\text{near}}$ .

The next sequence of inequalities shows a bound similar to (10), but when the sum is taken over all points of  $L$  (instead of only the points of  $L_{\text{far}}$ , as in Equation (10)).

$$\begin{aligned} \sum_{p \in L} \text{dist}_r(p, c^*) &\leq \sum_{p \in L} \text{dist}(p, c^*) = \sum_{p \in L_{\text{near}}} \text{dist}_r(p, c^*) + \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) \\ &= (1 + \frac{\delta}{4}) \cdot \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) && \text{(From (11))} \\ &\leq (1 + \frac{\delta}{4}) \cdot (1 + \frac{\delta}{4}) \cdot \sum_{p \in L_{\text{far}}} \text{dist}_r(p, c^*) && \text{(From (10))} \\ &\leq (1 + \frac{3\delta}{4}) \cdot \sum_{p \in L} \text{dist}_r(p, c^*) && (12) \end{aligned}$$

Where the last inequality follows from (i)  $(1 + \frac{\delta}{4}) \cdot (1 + \frac{\delta}{4}) \leq 1 + \frac{3\delta}{4}$  and (ii)  $L_{\text{far}} \subseteq L$ . This completes the proof of the claim.  $\triangleleft$

Using this claim, we prove the following claim, which shows that it is sufficient to find an approximate 1-median solution for  $L$ , which will also be a good approximation for HYBRID 1-MEDIAN for  $L$ . To this end, let  $\tilde{c}^* \in \mathbb{R}^d$  denote the optimal 1-median for  $L$ .

▷ Claim 12. Let  $c_1$  be an  $(1 + \frac{\delta}{8})$ -approximation for 1-MEDIAN for  $L$ , i.e.,  $\sum_{p \in L} \text{dist}(p, c_1) \leq (1 + \frac{\delta}{8}) \cdot \sum_{p \in L} \text{dist}(p, \tilde{c}^*)$ . Then, it is also a  $(1 + \delta)$ -approximation for HYBRID 1-MEDIAN for  $L$ , i.e.,

$$\sum_{p \in L} \text{dist}_r(p, c_1) \leq (1 + \delta) \cdot \sum_{p \in L} \text{dist}_r(p, c^*) \quad (13)$$

Proof. Let  $c_1, \tilde{c}^* \in \mathbb{R}^d$  as defined above. Then,

$$\begin{aligned}
\sum_{p \in L} \text{dist}_r(p, c_1) &\leq \sum_{p \in L} \text{dist}(p, c_1) \leq (1 + \frac{\delta}{8}) \cdot \sum_{p \in L} \text{dist}(p, \tilde{c}^*) && \text{(By definition of } c_1) \\
&\leq (1 + \frac{\delta}{8}) \cdot \sum_{p \in L} \text{dist}(p, c^*) \\
&\text{(Since } c_1 \in \mathbb{R}^d \text{ is an optimal 1-median and } c^* \in \mathbb{R}^d \text{ is a feasible median)} \\
&\leq (1 + \frac{\delta}{8}) \cdot (1 + \frac{3\delta}{4}) \cdot \sum_{p \in L} \text{dist}_r(p, c^*) && \text{(From Claim 11)} \\
&\leq (1 + \delta) \cdot \sum_{p \in L} \text{dist}_r(p, c^*) && \triangleleft
\end{aligned}$$

Thus, now the task reduces to finding a  $1 + \frac{\delta}{8}$ -approximate 1-MEDIAN solution for  $L$ . To this end, we have the following result from [24, 25].

► **Proposition 13** ([24, 25]). *Let  $X \subset \mathbb{R}^d$  be a set of  $n$  points and  $0 < \delta < 1$ . Let  $S \subseteq X$  be a uniform sample chosen from  $X$  of size  $\beta = (\frac{1}{\delta})^{c'}$ . Then, there exists an algorithm that runs in time  $2^{\mathcal{O}(1/\delta^c)d}$ , and with probability at least  $\alpha'$ , returns an  $(1 + \delta)$ -approximate 1-median for  $X$ . Here,  $c, c'$  are absolute constants independent of the dimension  $d$ .*

We combine the properties of the sample  $S_{q^*}$  proved in Claim 10 along with the previous proposition, to complete the proof. To this end, note that the first item of Claim 10 implies that, with probability at least  $1/2$ ,  $S_{q^*}$  contains at least  $\beta = \frac{1}{\delta^c}$  points from  $L$ . Then, we use the algorithm of Proposition 13, that returns with probability at least  $\alpha'$ , a  $(1 + \frac{\delta}{8})$ -approximate 1-median  $c_1 \in \mathbb{R}^d$  for  $L$ . It follows that,

$$\begin{aligned}
&\sum_{c \in F' \cup \{c_1\}} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o \setminus \{c^*\}} \text{cost}_r(\text{cl}(c, F), c) \\
&\leq \sum_{c \in F'} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) - \text{cost}_r(\text{cl}(c^*, F), c^*) + \sum_{p \in L} \text{dist}_r(p, c_1) \\
&\leq \sum_{c \in F'} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) - \text{cost}_r(\text{cl}(c^*, F), c^*) \\
&\quad + \delta \cdot \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) && \text{(From Claim 12)} \\
&\leq (1 + \delta) \cdot \left( \sum_{c \in F'} \text{cost}_{r'}(\text{cl}(c, F), c) + \sum_{c \in F_o} \text{cost}_r(\text{cl}(c, F), c) \right) \\
&\leq (1 + \delta)^{k-m+1} \cdot \text{OPT}_r && (14)
\end{aligned}$$

Then, by induction hypothesis,  $\text{HybridClustering}(F' \cup \{c_1\}, k, m - 1)$ , with probability at least  $\alpha^{m-1}$ , returns a solution  $\tilde{F}$  such that  $\text{cost}_{r'}(P, \tilde{F}) \leq (1 + \delta)^k \cdot \text{OPT}_r$ . The overall probability of this event is at least  $\frac{1}{2} \cdot \alpha' \cdot \alpha^{m-1} = \alpha^m$ , completing the induction.

This finishes the case analysis, and thus we have established the invariant using induction. Using the invariant, we can show the following key lemma.

► **Lemma 14.** *HybridClustering( $\emptyset, k, k$ ) returns a  $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation solution to the given instance of HYBRID  $k$ -CLUSTERING with probability at least  $\alpha^k$  for some constant  $0 < \alpha < 1$ .*

## 4:16 Hybrid $k$ -Clustering: Blending $k$ -Median and $k$ -Center

**Proof.** We first show the following:

$$\triangleright \text{Claim 15. } |R| \leq \left(\frac{k\sqrt{d}}{\delta}\right)^{\mathcal{O}(d)} + \frac{k \log n}{\delta^{\mathcal{O}(1)}} \cdot \left( \left(\frac{\sqrt{d}}{\delta}\right)^{\mathcal{O}(d)} + \binom{\beta'}{\beta} \right) \leq (\log n) \cdot 2^{(kd/\delta)^{\mathcal{O}(1)}}.$$

**Proof.** First, in Algorithm 1, we add the points returned by  $\text{Grid}(c', 16r, \delta r)$  for each  $c' \in F'$ . The number of such points is  $\left(\frac{16r\sqrt{d}}{\delta r}\right)^d = \left(\frac{\sqrt{d}}{\delta}\right)^{\mathcal{O}(d)}$ . Next, there are at most  $\frac{\log n}{\delta^{\mathcal{O}(1)}}$  values for  $q$  (this follows from the second preprocessing step, cf. Lemma 6), corresponding to each iteration of the for loop. In each iteration, we take a sample  $S$  of size  $\beta' = \mathcal{O}\left(\frac{150k\beta}{\delta^3}\right)$ . Then, for each  $p \in S$ , we add to  $R$  the points of  $\text{Grid}(p, 8r/\delta, \delta r)$ , and the number of such points is at most  $\left(\frac{8\sqrt{d}}{\delta^2}\right)^d = \left(\frac{\sqrt{d}}{\delta^2}\right)^{\mathcal{O}(d)}$ . In addition, we iterate over each subset  $S' \subseteq S$  of size  $\beta$ , and the number of such subsets is  $\binom{\beta'}{\beta} \leq \left(\frac{\epsilon\beta'}{\beta}\right)^\beta = \left(\frac{k}{\delta^3 \cdot \beta}\right)^\beta = \left(\frac{k}{\delta^3}\right)^{(1/\delta)^{\mathcal{O}(1)}} = k^{1/\delta^{\mathcal{O}(1)}}$ . Thus, overall, the size of  $R$  is bounded by  $(\log n) \cdot 2^{(kd/\delta)^{\mathcal{O}(1)}}$ .  $\triangleleft$

To bound the running time of the algorithm, let  $T(m)$  denote an upper bound on  $\text{HybridClustering}(F', k, m)$  for any  $F' \subset \mathbb{R}^d$ . Note that we make a recursive call on each point in  $R$ . Further by Proposition 13, the time taken to compute a center in Algorithm 1 is at most  $2^{(1/\delta)^{\mathcal{O}(1)}}$ ; and this algorithm is used in each of the at most  $\frac{k \log n}{(1/\delta)^{\mathcal{O}(1)}} \cdot \binom{\beta'}{\beta} \leq (\log n) \cdot k^{(1/\delta)^{\mathcal{O}(1)}}$ . Thus,  $T(m)$  can be bounded by the following recurrence.

$$\begin{aligned} T(m) &\leq |R| \cdot T(m-1) + (\log n) \cdot k^{(1/\delta)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)} \\ &\leq (\log n) \cdot 2^{\left(\frac{kd}{\delta}\right)^{\mathcal{O}(1)}} \cdot T(m-1) + k^{(1/\delta)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)} \end{aligned}$$

It can be shown that this recurrence solves to  $T(m) \leq 2^{\left(\frac{kd}{\delta}\right)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$  – here we use the standard argument that  $(\log n)^k \leq k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ .

Finally, note that our first call to the recursive algorithm is  $\text{HybridClustering}(F' = \emptyset, k, k)$ . At this point, the precondition of the invariant is satisfying setting  $F_o \leftarrow F^*$ , an optimal solution satisfying  $\text{cost}_r(P, F^*) = \text{OPT}_r$ . Then, the correctness of the invariant implies that, with probability at least  $\alpha^k$ , the algorithm returns a solution  $\tilde{F}$  of size at most  $k$ , that is a  $(1 + \epsilon, 1 + \epsilon)$ -bicriteria approximation – here we use that  $\delta = \frac{\epsilon}{10k}$ , which implies that  $(1 + \delta)^k \leq (1 + \epsilon)$ .  $\blacktriangleleft$

We now conclude with the following theorem, which is restated for convenience.

**► Theorem 2.** *Let  $0 < \epsilon < 1$ . There exists a randomized algorithm that, given an instance of HYBRID  $k$ -CLUSTERING in  $\mathbb{R}^d$ , runs in time  $2^{\left(\frac{kd}{\epsilon}\right)^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ , and returns a  $(1 + \epsilon, 1 + \epsilon)$ -approximation with probability at least a positive constant.*

**Proof.** From Lemma 14, the success probability of the algorithm is  $\alpha^k$  for some constant  $\alpha > 0$ . Thus, we need to repeat the algorithm  $\alpha^{-k}$  times to boost the probability to at least a positive constant, which gets absorbed in the  $2^{\left(\frac{kd}{\epsilon}\right)^{\mathcal{O}(1)}}$  factor.  $\blacktriangleleft$

### A Hybrid of $k$ -CENTER and $k$ -MEANS

We note that an almost identical algorithm also implies a  $(1 + \epsilon, 1 + \epsilon)$  bicriteria approximation for an analogous generalization of  $k$ -CENTER and  $k$ -MEANS. In this problem, the objective of (1) is replaced by the following:  $\text{cost}_r(C, F) := \sum_{p \in C} \text{dist}_r(p, C)^2$ . Let us refer to this problem as HYBRID  $(k, 2)$ -CLUSTERING – the “2” in the name refers to the squares of the

distance-thresholds that feature in the objective. Most of the analysis can be adapted to deal with the squares of the distances, by appropriately changing the sizes and distance-thresholds. The only significant change is that instead of Proposition 13, one needs to use an algorithm that computes an approximate 1-MEANS solution given a large enough uniform sample of the cluster – such an algorithm can also be found in [25]. Then, one obtains the following theorem.

► **Theorem 16.** *Let  $0 < \varepsilon < 1$ . There exists a randomized algorithm that, given an instance of HYBRID  $(k, 2)$ -CLUSTERING in  $\mathbb{R}^d$ , runs in time  $2^{(\frac{kd}{\varepsilon})^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$  and returns a  $(1 + \varepsilon, 1 + \varepsilon)$ -approximation with probability at least a positive constant.*

More generally, one can also define HYBRID  $(k, z)$ -CLUSTERING analogously, where the threshold-distances feature the  $z$ -th power of distances. Again, our approach easily extends to this problem, modulo a version of Proposition 13 for the vanilla  $(k, z)$ -CLUSTERING in Euclidean spaces. To the best of our knowledge, such an algorithm is not explicitly known in the literature; however, it may be possible to obtain such an algorithm using the approach of [23, 25].

### 3 Conclusion and Future Directions

In this paper, we proposed a novel clustering objective and defined a new problem, called HYBRID  $k$ -CLUSTERING, that generalizes both  $k$ -MEDIAN and  $k$ -CENTER. For  $d$ -dimensional euclidean inputs, we designed a randomized  $(1 + \varepsilon, 1 + \varepsilon)$ -bicriteria approximation scheme for HYBRID  $k$ -CLUSTERING running in time  $2^{(kd/\varepsilon)^{\mathcal{O}(1)}}$ , for any  $\varepsilon > 0$ . Further, essentially the same algorithm also generalizes for a hybrid objective of  $k$ -CENTER and  $k$ -MEANS. We remind that improving either of the two  $(1 + \varepsilon)$  factors to 1 would imply an exact FPT algorithm for  $k$ -CENTER/MEDIAN(/MEANS) in Euclidean spaces, which is unlikely to exist.

Our work opens up several interesting research directions. An immediate question is whether improving or removing the FPT dependence on the dimension  $d$  is possible, similar to the approach in [25] for  $k$ -MEDIAN/MEANS. One potential direction for achieving this could be the recent result that imports the famous Johnson-Lindenstrauss dimension reduction technique to  $k$ -clustering problems [26]. Another intriguing question is the design of coresets for HYBRID  $k$ -CLUSTERING, which could also have some implications for the previous problem via the approach of [9]. However, at a high level, designing coresets for HYBRID  $k$ -CLUSTERING appears to be challenging, since *a priori* we do not know which points belong inside the radius- $r$  balls (and thus contribute 0 to the cost), and which ones lie outside, and hence their cost needs to be approximately preserved.

Finally, considering HYBRID  $k$ -CLUSTERING with inputs from arbitrary metric spaces, a primal-dual algorithm from [7] can be adapted to obtain an  $(\alpha, \beta)$ -bicriteria approximation in polynomial time, for some constants  $\alpha$  and  $\beta$ <sup>3</sup>. Exploring the best possible constants in the bicriteria approximation would be an interesting avenue for future research.

---

#### References

- 1 Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation for robust clustering in discrete geometric spaces. *CoRR*, abs/2305.07316, 2023. doi:10.48550/arXiv.2305.07316.

---

<sup>3</sup> A quick examination of the proof of [7] suggests that  $(18, 6)$ -bicriteria approximation easily follows, with further improvements possible with more careful analysis.

- 2 Fateme Abbasi, Sandip Banerjee, Jaroslaw Byrka, Parinya Chalermsook, Ameet Gadekar, Kamyar Khodamoradi, Dániel Marx, Roohani Sharma, and Joachim Spoerhase. Parameterized approximation schemes for clustering with general norm objectives. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1377–1399. IEEE, 2023. doi:10.1109/FOCS57990.2023.00085.
- 3 Pankaj K. Agarwal, Sariel Har-Peled, and Hai Yu. Robust shape fitting via peeling and grating coresets. *Discret. Comput. Geom.*, 39(1-3):38–58, 2008. doi:10.1007/S00454-007-9013-2.
- 4 Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002. doi:10.1007/S00453-001-0110-Y.
- 5 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, (STOC)*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.
- 6 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 (STOC)*, pages 620–631. ACM, 2018. doi:10.1145/3188745.3188930.
- 7 Deeparnab Chakrabarty and Chaitanya Swamy. Interpolating between  $k$ -median and  $k$ -center: Approximation algorithms for ordered  $k$ -median. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPICs*, pages 29:1–29:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ICALP.2018.29.
- 8 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 (SODA)*, pages 642–651. ACM/SIAM, 2001. URL: <http://dl.acm.org/citation.cfm?id=365411.365555>.
- 9 Moses Charikar and Erik Waingarten. The johnson-lindenstrauss lemma for clustering and subspace approximation: From coresets to dimension reduction. *CoRR*, abs/2205.00371, 2022. doi:10.48550/arXiv.2205.00371.
- 10 Vincent Cohen-Addad, Arnaud de Mesmay, Eva Rotenberg, and Alan Roytman. The bane of low-dimensionality clustering. 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 441–456. SIAM, 2018. doi:10.1137/1.9781611975031.30.
- 11 Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for  $k$ -median and  $k$ -means. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *LIPICs*, pages 42:1–42:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ICALP.2019.42.
- 12 Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for  $k$ -means and  $k$ -median in euclidean and minor-free metrics. *SIAM J. Comput.*, 48(2):644–667, 2019. doi:10.1137/17M112717X.
- 13 Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for  $k$ -median and  $k$ -means coresets. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1038–1051. ACM, 2022. doi:10.1145/3519935.3519946.
- 14 Mark de Berg, Leyla Biabani, and Morteza Monemizadeh.  $k$ -center clustering with outliers in the MPC and streaming model. In *IEEE International Parallel and Distributed Processing Symposium, (IPDPS)*, pages 853–863. IEEE, 2023. doi:10.1109/IPDPS54959.2023.00090.
- 15 Hu Ding, Haikuo Yu, and Zixiu Wang. Greedy strategy works for  $k$ -center clustering with outliers and coresets construction. In *27th Annual European Symposium on Algorithms (ESA)*, volume 144 of *LIPICs*, pages 40:1–40:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ESA.2019.40.
- 16 Alon Efrat, Micha Sharir, and Alon Ziv. Computing the smallest  $k$ -enclosing circle and related problems. *Comput. Geom.*, 4:119–136, 1994. doi:10.1016/0925-7721(94)90003-5.

- 17 Fedor V. Fomin, Petr A. Golovach, Tanmay Inamdar, Saket Saurabh, and Meirav Zehavi. Hybrid k-clustering: Blending k-median and k-center, 2024. [arXiv:2407.08295](#).
- 18 Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *SIAM J. Comput.*, 48(2):452–480, 2019. doi:10.1137/17M1127181.
- 19 Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.
- 20 Sariel Har-Peled. How to get close to the median shape. *Comput. Geom.*, 36(1):39–51, 2007. doi:10.1016/J.COMGEO.2006.02.003.
- 21 Sariel Har-Peled and Soham Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3):147–157, 2005. doi:10.1007/S00453-004-1123-0.
- 22 Sariel Har-Peled and Yusu Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004. doi:10.1137/S0097539703427963.
- 23 Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A simple D 2-sampling based PTAS for k-means and other clustering problems. *Algorithmica*, 70(1):22–46, 2014. doi:10.1007/S00453-013-9833-9.
- 24 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In *32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*, pages 1374–1385. Springer, 2005. doi:10.1007/11523468\_111.
- 25 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010.
- 26 Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of Johnson-Lindenstrauss transform for  $k$ -means and  $k$ -medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1027–1038. ACM, 2019. doi:10.1145/3313276.3316350.
- 27 Dániel Marx. Efficient approximation schemes for geometric problems? In *13th Annual European Symposium on Algorithms (ESA)*, volume 3669 of *Lecture Notes in Computer Science*, pages 448–459. Springer, 2005. doi:10.1007/11561071\_41.
- 28 Jirí Matousek. On enclosing  $k$  points by a circle. *Inf. Process. Lett.*, 53(4):217–221, 1995. doi:10.1016/0020-0190(94)00190-A.
- 29 Nimrod Megiddo and Kenneth J Supowit. On the complexity of some common geometric location problems. *SIAM journal on computing*, 13(1):182–196, 1984.
- 30 Arie Tamir. The  $k$ -centrum multi-facility location problem. *Discret. Appl. Math.*, 109(3):293–307, 2001. doi:10.1016/S0166-218X(00)00253-5.
- 31 Hai Yu, Pankaj K. Agarwal, Raghunath Poreddy, and Kasturi R. Varadarajan. Practical methods for shape fitting and kinetic data structures using coresets. *Algorithmica*, 52(3):378–402, 2008. doi:10.1007/S00453-007-9067-9.





# Asynchronous Majority Dynamics on Binomial Random Graphs

Divyarthi Mohan  

Blavatnik School of Computer Science, Tel Aviv University, Israel

Paweł Prałat 

Department of Mathematics, Toronto Metropolitan University, Canada

---

## Abstract

We study information aggregation in networks when agents interact to learn a binary state of the world. Initially each agent privately observes an independent signal which is *correct* with probability  $\frac{1}{2} + \delta$  for some  $\delta > 0$ . At each round, a node is selected uniformly at random to update their public opinion to match the majority of their neighbours (breaking ties in favour of their initial private signal). Our main result shows that for sparse and connected binomial random graphs  $\mathcal{G}(n, p)$  the process stabilizes in a *correct* consensus in  $\mathcal{O}(n \log^2 n / \log \log n)$  steps with high probability. In fact, when  $\log n/n \ll p = o(1)$  the process terminates at time  $\hat{T} = (1 + o(1))n \log n$ , where  $\hat{T}$  is the first time when all nodes have been selected at least once. However, in dense binomial random graphs with  $p = \Omega(1)$ , there is an information cascade where the process terminates in the *incorrect* consensus with probability bounded away from zero.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Stochastic processes; Mathematics of computing  $\rightarrow$  Discrete mathematics; Theory of computation  $\rightarrow$  Social networks

**Keywords and phrases** Opinion dynamics, Social learning, Stochastic processes, Random Graphs, Consensus

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.5

**Category** APPROX

**Related Version** *Full Version:* [arXiv:2309.04691](https://arxiv.org/abs/2309.04691)

**Funding** *Divyarthi Mohan:* This project was funded in part by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement no. 866132) and by the Israel Science Foundation (grant no. 317/17).

**Acknowledgements** This work was done while the authors were visiting the Simons Institute for the Theory of Computing.

## 1 Introduction

Our opinions and actions we take as individuals are often influenced by both our private knowledge of the world and the information we obtain through our interactions with others. For example, a voter deciding which candidate’s economics policies would decrease inflation, might have an initial belief based on her own past expenditure and later might be swayed by her friends’ opinions. Now more than ever, with the advent of social media and online platforms, our interactions have increased many folds and our social networks are massive. Hence, an important research question is to understand if and how the structure of the social network and the dynamics of the interactions impact the (mis)information propagated [35]. Do our social networks enable successful information aggregation and lead to social learning, or do they amplify incorrect beliefs leading to an information cascade?

There has been extensive work modeling these opinion dynamics formally to study the network effects on information aggregation; see Section 1.4. In this paper, we focus on the model of *asynchronous majority dynamics*, where agents in a network (asynchronously) update



© Divyarthi Mohan and Paweł Prałat;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 5; pp. 5:1–5:20



Leibniz International Proceedings in Informatics

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

their opinions to match the majority opinion amongst their neighbours. In particular, each agent initially has a private belief over a binary state of the world and no publicly announced opinion. At each time step, an agent is chosen uniformly at random to announce/update her opinion and she does so by simply copying the majority of the neighbours' current announced opinions, breaking ties with her initial belief. Majority dynamics is clearly a naïve learning model, as the agents do not reason about potential information redundancy due to interaction between one's neighbours. Such naïve learning (non-Bayesian) models are a more faithful abstraction of everyday interactions between agents with bounded rationality (e.g., voters and consumers), while Bayesian models are a better abstraction of rational agents or interactions about high-stakes information (e.g., traders and scientists). We consider asynchronous updates which are more suitable to capture human decision making. Moreover, asynchronous emergence of announcements also captures an initial information diffusion phase before conventional social learning starts.

In our model, there is a *correct* opinion (i.e., the true state of the world) and each agent's initial private belief is independently drawn and is biased towards being correct (with probability  $1/2 + \delta$ ). So initially, in a large network, there is enough information so that an omniscient central planner can infer the true state (with very high probability). However, agents in the network are updating their opinions based on local heuristics, so the network structure can crucially alter the final outcome of the dynamics. For example, in a complete graph, with a constant probability all the nodes converge to the wrong opinion. On the other hand, in a star graph with high probability all the nodes converge to the correct opinion. This brings us to the main question of interest:

*“What network structures enable efficient social learning, where the dynamics stabilizes with every agent in the network reaching the correct opinion?”*

Feldman et al. [20], who initiated the study of asynchronous majority dynamics, showed that when the network is sparse (has bounded degree) and expansive, a correct consensus is reached with high probability. More recently, Bahrani et al. [4] studied networks that have certain tree structures (like preferential attachment trees and balanced  $m$ -ary trees) and showed that the dynamics stabilizes in a correct majority. Both results heavily rely on these particular assumptions on the network. For example, to even establish that a *majority of the nodes* have the correct opinion *at some point* in the process, it is crucial that the network is either a bounded degree graph or is a tree. In this paper, our goal is to extend the guarantees of asynchronous majority dynamics beyond these assumptions and to develop techniques applicable to more general networks formed through random graph models.

## 1.1 The Model

Consider any undirected graph  $G = (V, E)$  on  $n = |V|$  nodes. Individuals initially have one of two private *beliefs* which we will refer to as “Correct” (or 1) and “Incorrect” (or 0). Formally, each  $v \in V(G)$  receives an independent private signal  $X(v) \in \{0, 1\}$ , and  $\Pr(X(v) = 1) = 1/2 + \delta$ , for some universal constant  $\delta \in (0, 1/2)$ . Individuals also have a publicly announced *opinion* which we will simply refer to as an announcement or opinion. We define  $C^t(v) \in \{\perp, 0, 1\}$  to be the public announcement of  $v \in V$  at time  $t$ . Initially, no announcement have been made, that is,  $C^0(v) = \perp$  for all  $v \in V$ . In each subsequent step, a single node  $v^t$  is chosen uniformly at random from  $V$ , independently from the history of the process. In particular, as in the classical coupon collector problem, some nodes will be chosen many times before others will get lucky to get chosen for the first time. In step  $t$ ,  $v^t$  updates her announcement using *majority dynamics*, while announcements of other nodes stay the same. To be specific, for any  $i \in \{\perp, 0, 1\}$  and  $v \in V$ , let  $N_i^t(v)$  denotes the number of neighbours of  $v$  that have opinion  $i$  at time  $t$ . Then,

$$C^t(v) = \begin{cases} 1 & \text{if } N_1^{t-1}(v) > N_0^{t-1}(v) \text{ and } v = v^t, \\ 0 & \text{if } N_1^{t-1}(v) < N_0^{t-1}(v) \text{ and } v = v^t, \\ X(v) & \text{if } N_1^{t-1}(v) = N_0^{t-1}(v) \text{ and } v = v^t, \\ C^{t-1}(v) & \text{if } v \neq v^t. \end{cases}$$

Finally, for any  $i \in \{\perp, 0, 1\}$ , let  $Y_i^t$  be the number of nodes that have opinion  $i$  at time  $t$ , that is,  $Y_i^t = |\{v \in V : C^t(v) = i\}|$ .

As shown in [20], it is easy to see that in any network this process stabilizes with high probability in  $\mathcal{O}(n^2)$  steps. In fact, the process stabilizes in  $\mathcal{O}(n \log n + n \cdot d(G))$  where  $d(G)$  is the diameter of the graph [4]. That is, the network reaches a state at some time  $T$  where no node will want to change its announcement and thus the process terminates. Our goal is to understand what fraction of nodes converges to the correct opinion, that is, what the value of  $Y_1^T/n$  is.

## 1.2 Our Results

The main contribution of this paper is the proof that the asynchronous majority dynamics on *binomial random graph*  $\mathcal{G}(n, p)$  converges to the correct opinion, provided that the graph is sparse (that is, the average degree  $np = o(n)$ ) and connected (that is,  $np - \log n \gg 1$ ). If  $np \gg \log n$ , then the process converges to the correct opinion as quickly as it potentially could.

► **Theorem 1.** *Let  $\delta \in (0, 1/10]$ . Let  $\omega' = \omega'(n) = o(\log n)$  be any function that tends to infinity as  $n \rightarrow \infty$ . Suppose that  $p = p(n) \ll 1$  and  $p \gg \log n/n$ , and consider the asynchronous majority dynamics on  $\mathcal{G}(n, p)$ .*

*Then, asymptotically almost surely (a.a.s.) after  $n(\log n + \omega') = (1 + o(1))n \log n$  rounds the process terminates with all nodes announcing the correct opinion. In fact, it happens exactly at time  $\hat{T}$ , where  $\hat{T}$  is the first time when all nodes are selected at least once.*

For sparser (but still connected) graphs, the process also converges to the correct opinion. In this case, we do not aim to show that it happens at time  $\hat{T}$  and we only provide an upper bound for the number of rounds. It remains an open problem to determine if the process terminates at time  $\hat{T}$  or it needs more time to converge.

► **Theorem 2.** *Let  $\delta \in (0, 1/10]$ . Let  $\omega' = \omega'(n) = o(\log n)$  be any function that tends to infinity as  $n \rightarrow \infty$ . Suppose that  $p = p(n) \leq \omega' \log n/n$  and  $p \geq (\log n + \omega')/n$ , and consider the asynchronous majority dynamics on  $\mathcal{G}(n, p)$ .*

*Then, a.a.s. after  $\mathcal{O}(n(\log n)^2/(\log \log n))$  rounds the process terminates with all nodes announcing the correct opinion.*

These results are best possible in the following sense. If  $p \leq (\log n - \omega')/n$ , then a.a.s.  $\mathcal{G}(n, p)$  is disconnected. In fact, a.a.s. there are at least  $\omega'$  isolated nodes which announce their own private beliefs. As a result, a.a.s. some nodes announce the correct opinion but some of them announce the incorrect one. Indeed, the probability that all isolated nodes converge to the same opinion is at most  $o(1) + (1/2 + \delta/2)^{\omega'} + (1/2 - \delta/2)^{\omega'} = o(1)$ . On the other hand, if  $p \in (0, 1]$  is a constant separated from zero, then with positive probability the process converges to the correct opinion and with positive probability it converges to the incorrect opinion.

► **Theorem 3.** *Let  $\delta \in (0, 1/2)$ . Let  $\omega' = \omega'(n) = o(\log n)$  be any function that tends to infinity as  $n \rightarrow \infty$ . Suppose that  $p \in (0, 1]$  is a constant, and consider the asynchronous majority dynamics on  $\mathcal{G}(n, p)$ .*

*Then, the following is true for  $i \in \{0, 1\}$ : with probability at least  $p_i$ , after  $n(\log n + \omega') = (1 + o(1))n \log n$  rounds the process terminates with all nodes announcing opinion  $i$ , where*

$$p_1 = (1/2 + \delta) \exp\left(-\log(1/p)(1/p)\right) > 0$$

$$p_0 = (1/2 - \delta) \exp\left(-\log(1/p)(1/p)\right) > 0.$$

Finally, let us mention that for some technical reason, in Theorems 1 and 2 it is assumed that  $\delta \leq 1/10$ . However, it is easy to couple the process with  $\delta \leq 1/10$  with the one with  $\delta \in (1/10, 1/2)$  to show that the result holds for any  $\delta \in (0, 1/2)$  – see Subsection 2.3 for more details.

### 1.3 Future Directions

Let us highlight a few potential directions one might want to consider.

- As already mentioned above, for very sparse graphs ( $np - \log n \rightarrow \infty$  and  $np = \mathcal{O}(\log n)$ ), it would be interesting to determine if the process terminates at time  $\hat{T}$  or it needs more time to converge to the correct opinion – see Theorem 2.
- Theorem 2 holds as long as  $pn = \log n + \omega$  for some  $\omega = \omega(n) \rightarrow \infty$  as  $n \rightarrow \infty$ . It is known that if  $pn = \log n + c$  for some constant  $c \in \mathbb{R}$ , then with probability bounded away from one and from zero the graph is disconnected. As a result, there is no hope to extend the result for this range of  $p$ . But it is plausible that a.a.s. it holds right at the time the random graph process creates a connected graph. This would be an optimal “hitting time” result.
- For disconnected graphs ( $np - \log n \rightarrow -\infty$ ), it would be interesting to investigate the process run on the giant component of  $\mathcal{G}(n, p)$ .
- For dense graphs, it is not true that a.a.s. all nodes converge to the correct opinion – see Theorem 3. Having said that, it is reasonable to expect that a.a.s. all nodes converge to the same opinion (for example, [21] show that a consensus is reached in this case in a synchronous setting). Is it true in our asynchronous setting? In any case, what is the asymptotic value of the probability that all nodes converge to the correct opinion?
- It would be interesting to investigate other random graph models that are able to generate graphs with power-law degree distributions as the Chung-Lu model [16] or the classical configuration model. More challenging, but an important and interesting, direction would be to understand the learning process on a network with a community structure such as the ABCD (Artificial Benchmark for Community Detection) model [30] which produces a random graph with community structure and power-law distribution for both degrees and community sizes. In this model, small communities might create echo chambers, environments in which participants encounter beliefs that amplify or reinforce their preexisting beliefs inside a community and insulated from rebuttal.

### 1.4 Related Work

In this section, we briefly discuss prior work on social learning mainly focusing on the setting with a binary state of the world and the agents initially have a correct opinion independently with probability  $1/2 + \delta$ . We refer to some recent surveys on social learning and opinion dynamics [37, 8, 11] for a more detailed literature review.

Majority dynamics falls under a wide class of naive or non-Bayesian models, where agents use a simple local heuristic to update their opinions, to capture simple behaviours exhibited by non-expert decision makers. Prior works have studied majority dynamics under a variety of modeling assumptions, to understand when a consensus is possible and when there is social learning – that is, the consensus (or the majority) is correct. These works study a variety of networks such as  $k$ -regular trees [28, 32, 4], bounded degree graphs [20], random regular graphs [23], “symmetric” graphs and expanders [39]. In [44], a different perspective on social learning asks when is it possible to “recover the correct opinion” at the end of the dynamics through any function (not just a consensus or majority vote). Prior work has also considered models with different notions of bias towards correct opinion, for example, each node updates to the correct opinion with some probability [3], or the initial configuration of the network has some  $n/2 + \delta$  correct opinions [45, 46].

Recently, there has been a series of work studying synchronous majority dynamics in binomial random graphs [10, 21, 15], with a focus to showing that 99% of the nodes converge to the same opinion (with high probability) for sparse random graphs, with  $p = \Omega(\log n/n^{3/5})$  being the best known lower bound for the average degree. Moreover, [47] showed that a correct consensus is reached with high probability for binomial random graphs with  $p = \Omega(\log n/n)$ . In contrast to these works, we focus on asynchronous dynamics and prove that a correct consensus is reached with high probability for  $p = \Omega(\log n/n)$  and  $p = o(1)$ . Binomial random graphs are also studied under label propagation [34] which is a special case of synchronous majority dynamics with non-binary opinion in  $[0, 1]$ .

Many of the works mentioned above focus on synchronous updates, where all agents update their opinions synchronously in each round. Majority dynamics with synchronous updates leads to a correct consensus for all networks that are sufficiently connected [39], whereas with asynchronous updates the network structure can have a huge impact on social learning. This is best illustrated by the complete graph. With asynchronous updates, once the first agent announces their opinion (which can be wrong with probability  $1/2 - \delta$ ) everyone will copy this. Hence, with a probability bounded away from zero all the nodes converge to the wrong opinion. In contrast, if all agents were to update synchronously, then the majority of the round one updates will be correct with high probability, so there will be a correct consensus in round two. Recent work [5], studies the DeGroot model with uninformed agents, to capture the different phases of information diffusion and social learning, which is a key phenomena that occurs in our asynchronous model.

Other non-Bayesian dynamics have also been extensively studied. In the Voter model, agents choose a random neighbour and copy their opinion [17, 27]. A similar dynamics called  $k$ -majority model are studied in the distributed computing literature, where agents choose  $k$ -neighbours at random and copy their majority [9, 25, 24, 18, 1]. In the DeGroot Model, an agent’s opinion lies in  $[0, 1]$  (as opposed to binary  $\{0, 1\}$ ) and agents update to the average of their neighbours [19, 26]. A key difference between these works and majority dynamics is that in these models a consensus is reached with probability 1 for any connected graphs. This is not the case in majority dynamics even with synchronous updates.

While our focus is in non-Bayesian dynamics, there has also been a long line of work studying Bayesian models, where agents update their beliefs rationally given their (local) observations exhibiting more sophisticated decision-making. Seminal works [7, 12] introduced the study of Bayesian dynamics and identified conditions that lead to information cascades. Here, the agents arrive sequentially and observe all the announcements (i.e., they form a complete graph), and many other subsequent works consider Bayesian dynamics under different assumptions and variations [43, 6, 14]. Bayesian dynamics in general social networks were first studied in [2]. There is also a long line of work studying Bayesian learning with repeated interactions [22, 42, 31, 41, 40, 38].

## 2 Preliminaries

### 2.1 Notation

Let us first precisely define the  $\mathcal{G}(n, p)$  binomial random graph.  $\mathcal{G}(n, p)$  is a distribution over the class of graphs with the set of nodes  $[n] := \{1, \dots, n\}$  in which every pair  $\{i, j\} \in \binom{[n]}{2}$  appears independently as an edge in  $G$  with probability  $p$ . Note that  $p = p(n)$  may (and usually does) tend to zero as  $n$  tends to infinity. We say that  $\mathcal{G}(n, p)$  has some property *asymptotically almost surely* or a.a.s. if the probability that  $\mathcal{G}(n, p)$  has this property tends to 1 as  $n$  goes to infinity. For more about this model see, for example, [13, 29, 33].

Given two functions  $f = f(n)$  and  $g = g(n)$ , we will write  $f(n) = \mathcal{O}(g(n))$  if there exists an absolute constant  $c \in \mathbb{R}_+$  such that  $|f(n)| \leq c|g(n)|$  for all  $n$ ,  $f(n) = \Omega(g(n))$  if  $g(n) = \mathcal{O}(f(n))$ ,  $f(n) = \Theta(g(n))$  if  $f(n) = \mathcal{O}(g(n))$  and  $f(n) = \Omega(g(n))$ , and we write  $f(n) = o(g(n))$  or  $f(n) \ll g(n)$  if  $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$ . In addition, we write  $f(n) \gg g(n)$  if  $g(n) = o(f(n))$  and we write  $f(n) \sim g(n)$  if  $f(n) = (1 + o(1))g(n)$ , that is,  $\lim_{n \rightarrow \infty} f(n)/g(n) = 1$ .

### 2.2 Concentration Tools

In this section, we state a few specific instances of Chernoff's bound that we will find useful. Let  $(Z_1, \dots, Z_n)$  be a sequence of independent Bernoulli( $p$ ) random variables. For each  $j \in [n]$ , let  $X_j = \sum_{i=1}^j Z_i$ . In particular,  $X_n \in \text{Bin}(n, p)$  is a random variable distributed according to a Binomial distribution with parameters  $n$  and  $p$ . Then, a consequence of *Chernoff's bound* (see e.g. [29, Theorem 2.1]) is that for any  $t \geq 0$  we have

$$\mathbb{P}(X_n - \mathbb{E}[X_n] \geq t) \leq \exp\left(-\frac{t^2}{2(\mathbb{E}[X_n] + t/3)}\right) \quad (1)$$

$$\mathbb{P}(\mathbb{E}[X_n] - X_n \geq t) \leq \exp\left(-\frac{t^2}{2\mathbb{E}[X_n]}\right). \quad (2)$$

Moreover, let us mention that the above bounds hold in a more general setting as well, that is, for any sequence  $(Z_j)_{1 \leq j \leq n}$  of independent random variables such that for every  $j \in [n]$  we have  $Z_j \in \text{Bernoulli}(p_j)$  with (possibly) different  $p_j$ -s (again, see e.g. [29] for more details).

Finally, we note that  $X_n - \mathbb{E}[X_n]$  in (1) can be replaced with  $\max_{1 \leq j \leq n} (X_j - \mathbb{E}[X_j])$  and  $\mathbb{E}[X_n] - X_n$  in (2) can be replaced with  $\max_{1 \leq j \leq n} (\mathbb{E}[X_j] - X_j)$ . That is, we have

$$\mathbb{P}\left(\max_{1 \leq j \leq n} (X_j - \mathbb{E}[X_j]) \geq t\right) \leq \exp\left(-\frac{t^2}{2(\mathbb{E}[X_n] + t/3)}\right) \quad (3)$$

$$\mathbb{P}\left(\max_{1 \leq j \leq n} (\mathbb{E}[X_j] - X_j) \geq t\right) \leq \exp\left(-\frac{t^2}{2\mathbb{E}[X_n]}\right). \quad (4)$$

This is a consequence of a standard martingale bound (see e.g. [36] for more details).

### 2.3 Coupling

Suppose that at some point of the process, the public announcement is captured by  $C^t(v)$ ,  $v \in V$ . Let  $\hat{C}^t(v)$  be any sequence of opinions such that the following properties hold: (a) if  $\hat{C}^t(v) = 1$ , then  $C^t(v) = 1$ , (b) if  $\hat{C}^t(v) = 0$ , then  $C^t(v) \in \{0, 1, \perp\}$ , (c) if  $\hat{C}^t(v) = \perp$ , then  $C^t(v) = \perp$ . In other words, we get the auxiliary sequence  $\hat{C}^t(v)$  by modifying some of the opinions 1 and  $\perp$  in  $C^t(v)$  to 0. Hence, the process starting from  $C^t(v)$  can be coupled with the auxiliary process starting from  $\hat{C}^t(v)$  such that all the properties (a)–(c) are satisfied in

every step of the process. In particular, if the auxiliary process converges to all nodes having opinion 1, then so does the original process. This easy observation will turn out to be useful in analyzing the process.

Similarly, suppose private beliefs in the auxiliary process are dominated by private beliefs in the original process: for any  $v \in V$ ,  $\hat{X}(v) \leq X(v)$ . If the two processes are coupled, then properties (a)–(c) hold again. As before, if the auxiliary process converges to all nodes having opinion 1, then so does the original process. In particular, as mentioned above, the assumption that  $\delta \in (0, 1/10]$  in Theorems 1 and 2 can be relaxed to  $\delta \in (0, 1/2)$ .

### 3 Sparse Random Graphs

In this section, we consider sparse random graphs, that is, we will assume that  $p = o(1)$ . Let  $\omega = \omega(n)$  be a function that tends to infinity as  $n \rightarrow \infty$ , arbitrarily slowly. In particular, each time we refer to  $\omega$ , we will assume that  $\omega \ll pn$  and  $\omega \ll (1/p)^{1/2}$  so that  $1/p \gg 1/(p\omega) \gg 1/(p\omega^2) \gg 1$ .

We will consider a few phases. During the first phase (Subsection 3.1), most of the nodes that are chosen have not yet announced their opinions ( $C^{t-1}(v^t) = \perp$ ) and none of their neighbours have announced ( $N_1^{t-1}(v^t) = N_0^{t-1}(v^t) = 0$ ). Hence, the announcement of  $v^t$  will typically coincide with its private belief. Moreover, most of the nodes selected will not be chosen again during this phase. During the second phase (Subsection 3.2), it is still the case that most selected nodes are selected for the first time but this time they might have neighbours that announced their opinions. As a result, the argument is more involved but the conclusion is that at the end of the second phase more nodes have correct opinion than not.

The analysis of the first two phases can be applied for all sparse graphs, even below the threshold for connectivity. The analysis of the final steps of the process is slightly more involved. We first present an easy argument for not very sparse graphs (Subsection 3.3), that is, when the asymptotic expected degree satisfies  $pn \gg \log n$ . Very sparse graphs for which  $pn = \Theta(\log n)$  (but, of course, above the connectivity threshold) are considered in Subsection 3.4.

#### Overview

A key phenomena in asynchronous dynamics is that the process involves both information diffusion and conventional social learning. Intuitively, the process initially produces some independent beliefs/opinions pop up sporadically throughout the network. These opinions then diffuses in the network during the process as more nodes are selected to announce/update their opinion by learning from their neighbours. With this in mind, our analysis considers multiple phases of the process. We provide a brief description of the different phases below.

- **Phase 1.** In the first few time steps, most nodes that are selected to announce have not been selected earlier and, more importantly, do not have neighbours who have been selected before. So almost all of the opinions in the network at the end of phase one are just the independent private beliefs of the selected nodes. Since the private signals are biased towards being correct, a strict majority of the opinions are correct at the end of the first phase. In particular, we show that at time  $T_1 = \delta/2p$ , the number of nodes with opinion 1 is at least  $(1/2 + 3\delta/5)T_1$  and opinion 0 is represented at most  $(1/2 - 3\delta/5)T_1$  times. Moreover,  $T_1(1 - o(1))$  nodes have made some announcement in this phase, that is, very few nodes were selected more than once.

- **Phase 2.** In the second phase, again most nodes that are selected to announce have not been selected earlier. In particular, we show that at any time  $t$  during the second phase (i.e., after time  $T_1$  but before time  $T_2 = n/\omega$ ), the number of nodes that were selected twice before time  $t$  is  $o(t)$ . Moreover, since a *super majority* of the opinions at the end of the previous phase were correct, we prove that nodes that are selected to announce for the first time are more likely to learn the correct opinion even if we pretend that the few nodes that are selected *again* were to change their opinion to 0.
- **Phase 3 (a).** For not very sparse graphs, we are able to show all nodes which were not selected in the first two phases have more neighbours with opinion 1 than not. Again, very few nodes who were selected before are selected again before time  $T_3 = n/\sqrt{\omega}$ , so even if all of them announce 0 all nodes who make their first announcement between time  $T_2$  and time  $T_3$  announce the correct opinion. Finally, even if all the nodes that were selected before time  $T_2$  are to have opinion 0 and all nodes that were selected for the first time between time  $T_2$  and time  $T_3$  have opinion 1, we show that a.a.s. all announcements after time  $T_3$  are always correct.
- **Phase 3 (b).** For very sparse graphs, the proof of the last phase is more involved as there might be nodes whose degree is too small to guarantee that a majority of their neighbours have opinion 1, even though there is a super majority of opinion 1 in the network. However, we may bound the number of nodes with small degrees and show that no large degree node has more than one small degree neighbour. With this in hand, we show that after every batch of  $O(n \log n)$  many time steps the number of large degree nodes with opinion 0 shrinks by at least  $(\log \log n)^{1/4}$  factor. Hence, after  $o(\log n)$  many such batches all large nodes have opinion 1. Finally, we show that no two small degree nodes are adjacent to each other, and hence all the small degree nodes will also switch to opinion 1 by copying the opinions of their large degree neighbours.

We highlight a few simple techniques that help us in the analysis. Firstly, separating the randomness of the graph, the node selection process and the opinion formation. For example, we wait to reveal the edges adjacent to a node only when she is selected to announce for the first time. Second, considering an auxiliary dynamics that is coupled with the actual dynamics in order to ignore problematic but rare events such as the repeated nodes in the first two phases. Finally, finding independent sequences of random variables that stochastically dominate the opinion dynamics sequence in order to compute probability bounds more easily.

### 3.1 Phase 1: $T_1 = \delta/(2p)$

In the analysis of the process, it will be convenient to ignore opinions of a small fraction of nodes, and consider the following *auxiliary dynamics*. We will use  $D^t(v) \in \{\perp, ?, 0, 1\}$  to denote the *auxiliary announcement* of  $v \in V$  at time  $t$ . For any  $i \in \{\perp, ?, 0, 1\}$ , let  $Z_i^t$  be the number of nodes that have auxiliary opinion  $i$  at time  $t$ , that is,  $Z_i^t = |\{v \in V : D^t(v) = i\}|$ . We will explain how the values of  $D^t(v)$  are determined soon but the auxiliary dynamics will be coupled with the original one and, in particular, we will make sure that the following property holds.

► **Property 4.** *If  $D^t(v) = i$  for some  $i \in \{\perp, 0, 1\}$  and time  $t$ , then  $C^t(v) = D^t(v)$ . On the other hand, if  $D^t(v) = ?$ , then  $C^t(v) \in \{0, 1\}$ . As a result, for  $i \in \{0, 1\}$  and any time  $t$  during the first phase, we have*

$$Z_i^t \leq Y_i^t \leq Z_i^t + Z_?^t. \quad (5)$$



The first phase takes  $T_1 = \delta/(2p) = \Theta(1/p) \gg \omega^2 \gg 1$  rounds. In order to keep the analysis easy, we postpone exposing edges of  $\mathcal{G}(n, p)$  for as long as possible, and keep the following useful property.

► **Property 5.** *At any time  $t$ , only edges of  $\mathcal{G}(n, p)$  with both endpoints in the set  $\{v : D^t(v) \neq \perp\}$  are exposed.*

The auxiliary dynamics, coupled with the original one, that we aim to understand is defined as follows. Consider a node  $v^t$  chosen at time  $t$ . For all other nodes  $v \neq v^t$  we have  $D^t(v) = D^{t-1}(v)$ . For  $v^t$  we have,

$$D^t(v^t) = \begin{cases} ? & \text{if } D^{t-1}(v^t) \neq \perp, \\ ? & \text{if } \exists \text{ node } v \text{ such that } v \in N(v^t) \text{ and } D^{t-1}(v) \neq \perp, \\ X(v^t) & \text{otherwise.} \end{cases}$$

That is, if  $v^t$  had announced her opinion at least once before time  $t$  ( $D^{t-1}(v^t), C^{t-1}(v^t) \neq \perp$ ), then we fix  $D^t(v^t) = ?$ . On the other hand, if  $v^t$  has not announced her opinion yet (that is,  $D^{t-1}(v^t) = C^{t-1}(v^t) = \perp$ ), then we expose edges of  $\mathcal{G}(n, p)$  between  $v^t$  and the set  $\{v : D^{t-1}(v) \neq \perp\}$ . If no edge between  $v^t$  and the set  $\{v : D^{t-1}(v) \neq \perp\}$  is present, then no neighbour of  $v^t$  has an announced opinion and so  $D^t(v^t) = C^t(v^t) = X(v^t)$  is fixed to the private belief of  $v^t$ . Otherwise (that is, at least one edge is present), then we simply fix  $D^t(v^t) = ?$ . Let us note that, an alternative approach would be to investigate the value of  $C^t(v^t)$  and then fix  $D^t(v^t) = C^t(v^t)$ . However, we expect at most  $pt \leq pT_1 = \delta/2$  edges between  $v^t$  and  $\{v : D^{t-1}(v) \neq \perp\}$ , and so there will not be many nodes  $v^t$  of this type. As a result, we may simply ignore the announcements of such nodes, thus simplifying our analysis.

Moreover, a useful implication of this approach is that in order to estimate the values of  $Z_{\perp}^t$  and  $Z_{?}^t$  in this process, we do not need to uncover nodes' private beliefs ( $X(v)$ 's). Hence, we may postpone exposing private beliefs of nodes with  $D^t(v) \notin \{\perp, ?\}$  to the very end of this phase, and only then expose this information to determine how many nodes satisfy  $D^{T_1}(v) = 1$  and how many of them satisfy  $D^{T_1}(v) = 0$ . Finally, it is easy to see that Property 4 is satisfied at time  $T_1$  and Property 5 is satisfied in any point of the process.

Here is the main result of this subsection.

► **Proposition 6.** *Suppose that  $p = p(n) \ll 1$  and  $p \gg 1/n$ . Set  $T_1 = \delta/(2p)$ . Let  $\omega = \omega(n) \ll \min\{pn, (1/p)^{1/2}\}$  be any function that tends to infinity as  $n \rightarrow \infty$ . Then, a.a.s. the following holds:*

$$Z_{?}^{T_1} \leq \frac{\delta T_1}{4} (1 + \mathcal{O}(1/\omega)) \quad (6)$$

$$Z_{\perp}^{T_1} \geq (1/2 + 3\delta/5) T_1 \quad (7)$$

$$Z_{?}^{T_1} + Z_{\perp}^{T_1} + Z_0^{T_1} = T_1 (1 - \mathcal{O}(1/\omega)). \quad (8)$$

As a result, by Property 4,

$$Y_1^{T_1} \geq (1/2 + 3\delta/5) T_1$$

$$Y_0^{T_1} \leq (1/2 - 3\delta/5) T_1$$

$$Y_1^{T_1} + Y_0^{T_1} = T_1 (1 - \mathcal{O}(1/\omega)).$$

**Proof.** Let us start with investigating  $Z_{?}^{T_1}$ . Recall that in our auxiliary dynamics, there are two ways node  $v^t$  could change its state to  $D^t(v^t) = ?$  at time  $t$ . Let  $I_t$  be the indicator random variable that this happens because  $D^{t-1}(v^t) \neq \perp$ , and let  $I = \sum_{t=1}^{T_1} I_t$ . Similarly, let  $J_t$  be the indicator random variable that  $D^{t-1}(v^t) = \perp$  but there is an edge between  $v^t$  and the set  $\{v : D^{t-1}(v) \neq \perp\}$ . Let  $J = \sum_{t=1}^{T_1} J_t$ .

## 5:10 Asynchronous Majority Dynamics on Binomial Random Graphs

Note that, at most  $t - 1$  distinct nodes have made an announcement before round  $t$ . In particular, at most one node can change its state from  $D^{t-1}(v) = \perp$  to  $D^t(v) \neq \perp$ , deterministically, at any round  $t$  of the process. So, the number of nodes with  $D^{t-1}(v) \neq \perp$  is  $n - Z_{\perp}^{t-1} \leq t - 1$ . We get that

$$\Pr(I_t = 1) = \frac{n - Z_{\perp}^{t-1}}{n} \leq \frac{t-1}{n},$$

and so  $I$  can be stochastically upper bound by  $\hat{I} = \sum_{t=1}^{T_1} \hat{I}_t$  where  $(\hat{I}_t)_{1 \leq t \leq T_1}$  are independent variables and for every  $t \in [T_1]$  we have  $\hat{I}_t \in \text{Bernoulli}((t-1)/n)$ . Note that, since  $pn \gg \omega$ ,

$$\mathbb{E}[\hat{I}] = \sum_{t=1}^{T_1} \frac{t-1}{n} = \frac{(T_1-1)T_1}{2n} \sim \frac{\delta T_1}{4pn} \ll \frac{T_1}{\omega}. \quad (9)$$

It follows from Chernoff's bound (Eq. (1)) (and the comment right after it) applied with  $t = T_1/\omega = \Theta(1/(p\omega)) \gg \omega \gg 1$  that

$$\Pr(\hat{I} \geq \mathbb{E}[\hat{I}] + t) \leq \exp\left(-\frac{t^2}{(2/3 + o(1))t}\right) = \exp(-\Theta(t)) = o(1).$$

So a.a.s.  $I \leq \hat{I} = \mathcal{O}(T_1/\omega)$ . Similarly, since  $pt \leq pT_1 = \delta/2 < 1/4$ ,

$$\Pr(J_t = 1) = \frac{Z_{\perp}^{t-1}}{n} \left(1 - (1-p)^{n-Z_{\perp}^{t-1}}\right) \leq 1 - (1-p)^t = 1 - \left(1 - pt + p^2 \binom{t}{2} - \dots\right) \leq pt.$$

As before, we stochastically upper bound  $J$  by  $\hat{J} = \sum_{t=1}^{T_1} \hat{J}_t$ , where  $\hat{J}_t \in \text{Bernoulli}(pt)$ . We get that

$$\mathbb{E}[\hat{J}] = \sum_{t=1}^{T_1} pt = \frac{p(T_1+1)T_1}{2} = \frac{pT_1^2}{2} (1 + \mathcal{O}(1/T_1)) = \frac{\delta T_1}{4} (1 + \mathcal{O}(1/\omega)),$$

and Chernoff's bound (Eq. (1)) (applied with  $t = \mathbb{E}[\hat{J}]/\omega$ ) implies that

$$\Pr(\hat{J} \geq \mathbb{E}[\hat{J}] + t) \leq \exp\left(-\frac{\mathbb{E}[\hat{J}]}{(2 + o(1))\omega^2}\right) = \exp(-\Theta(T_1/\omega^2)) = \exp(-\Theta(1/(p\omega^2))) = o(1).$$

Hence, a.a.s.  $J \leq \hat{J} \leq \frac{\delta T_1}{4} (1 + \mathcal{O}(1/\omega))$  and so a.a.s.  $Z_7^{T_1} \leq I + J \leq \frac{\delta T_1}{4} (1 + \mathcal{O}(1/\omega))$ . This proves (6).

It remains to investigate  $Z_0^{T_1}$  and  $Z_1^{T_1}$ . Let us summarize the situation at time  $T_1$ . The number of rounds when nodes were not chosen for the first time is at most  $I = \mathcal{O}(T_1/\omega)$  a.a.s. Hence, a.a.s. the number of nodes that were chosen at least once is equal to  $T_1 - \mathcal{O}(T_1/\omega)$ . This proves (8). Moreover, it implies that a.a.s. the number of nodes with  $D^{T_1}(v) \notin \{\perp, ?\}$  is equal to

$$Z_1^{T_1} + Z_0^{T_1} = T_1 - \mathcal{O}(T_1/\omega) - Z_7^{T_1} \geq (1 - \delta/4)T_1 (1 + \mathcal{O}(1/\omega)).$$

More importantly, as mentioned above, in the analysis so far we did not use their opinions which are consistent with their private beliefs. We conveniently deferred this information up to now. After exposing this information, we get that  $Z_1^{T_1}$  is stochastically lower bounded by the random variable  $\hat{Z}_1 \in \text{Bin}((1 - \delta/4)T_1 - cT_1/\omega, 1/2 + \delta)$ , where  $c > 0$  is a large enough constant. After applying Chernoff's bound (Eq. (2)) (with  $t = T_1/\omega$ ) we get that

$$\begin{aligned} Z_1^{T_1} \geq \hat{Z}_1 &= (1/2 + \delta)(1 - \delta/4)T_1(1 + \mathcal{O}(1/\omega)) \\ &\geq (1/2 + \delta - \delta/4)T_1(1 + \mathcal{O}(1/\omega)) \\ &\geq (1/2 + 3\delta/5)T_1 \end{aligned}$$

with probability at least

$$1 - \exp(-\Theta(T_1/\omega^2)) = 1 - \exp(-\Theta(1/(p\omega^2))) = 1 - o(1).$$

This proves (7).

The conclusion for  $Y_1^{T_1}$  follows immediately from Property 4, and the bound for  $Y_0^{T_1}$  is a trivial implication of the fact that  $Y_1^{T_1} + Y_0^{T_1} \leq T_1$ . The proof of the proposition is finished.  $\blacktriangleleft$

### 3.2 Phase 2: $T_2 = T_2(n)$ such that $\omega/p \leq T_2 \leq n/\omega$

By Proposition 6, since we aim for a statement that holds a.a.s., we may assume that at the beginning of Phase 2,

$$\begin{aligned} Y_1^{T_1} &\geq (1/2 + 3\delta/5) T_1 \\ Y_0^{T_1} &\leq (1/2 - 3\delta/5) T_1 \\ Y_1^{T_1} + Y_0^{T_1} &= T_1 (1 + \mathcal{O}(1/\omega)). \end{aligned}$$

As in the previous phase, it will be convenient to ignore opinions of some problematic nodes and assign auxiliary announcements  $D^t(v) = ?$  to such nodes. We will continue using  $Z_i^t$  to denote the number of nodes that have auxiliary opinion  $i$  at time  $t$ . We fix  $D^{T_1}(v) = C^{T_1}(v)$  for all  $v$  so, initially, auxiliary announcements coincide with the truth announcements. However, this time we assign  $D^t(v^t) = ?$  only if  $D^{t-1}(v^t) \neq \perp$  (that is, the node chosen at time  $t$  has made an announcement in the past); otherwise, the auxiliary announcement  $D^t(v^t)$  is determined immediately pretending that all neighbours  $v$  of  $v^t$  with  $D^{t-1}(v) = ?$  announced 0. More formally, for each node  $v$  and  $i \in \{0, 1, \perp, ?\}$  let  $\hat{N}_i^t(v)$  denote the number of neighbours  $v'$  of  $v$  with auxiliary opinion  $D^t(v') = i$  at time  $t$ . Then we have,

$$D^t(v^t) = \begin{cases} ? & \text{if } D^{t-1}(v^t) \neq \perp, \\ 1 & \text{if } \hat{N}_1^{t-1}(v^t) > \hat{N}_0^{t-1}(v^t) + \hat{N}_?^{t-1}(v^t), \\ 0 & \text{if } \hat{N}_1^{t-1}(v^t) < \hat{N}_0^{t-1}(v^t) + \hat{N}_?^{t-1}(v^t), \\ X(v^t) & \text{if } \hat{N}_1^{t-1}(v^t) = \hat{N}_0^{t-1}(v^t) + \hat{N}_?^{t-1}(v^t). \end{cases}$$

As a consequence,  $D^t(v)$  and  $C^t(v)$  are coupled so that the following property is satisfied.

► **Property 7.** *If  $D^t(v) = i$  for some  $i \in \{\perp, 1\}$  and time  $t$ , then  $C^t(v) = D^t(v)$ . On the other hand, if  $D^t(v) = i$  for some  $i \in \{0, ?\}$ , then  $C^t(v) \in \{0, 1\}$ . As a result, for any time  $t$  during the second phase, we have  $Y_1^t \geq Z_1^t$ .*

As before, it is easy to see that Property 5 is also satisfied during this phase. Here is the main result of this subsection.

► **Proposition 8.** *Suppose that  $p = p(n) \ll 1$  and  $p \gg 1/n$ . Let  $\omega = \omega(n) \ll \min\{pn, (1/p)^{1/2}\}$  be any function that tends to infinity as  $n \rightarrow \infty$ . Set  $T_2 = T_2(n)$  such that  $\omega/p \leq T_2 \leq n/\omega$ . Then, a.a.s. the following holds:*

$$\begin{aligned} Z_?^{T_2} &= \mathcal{O}(T_2/\omega) \\ Z_1^{T_2} &\geq (1/2 + \delta/2) T_2 \\ Z_?^{T_2} + Z_1^{T_2} + Z_0^{T_2} &= T_2 (1 - \mathcal{O}(1/\omega)). \end{aligned}$$

## 5:12 Asynchronous Majority Dynamics on Binomial Random Graphs

As a result, by Property 7,

$$\begin{aligned} Y_1^{T_2} &\geq (1/2 + \delta/2) T_2 \\ Y_0^{T_2} &\leq (1/2 - \delta/2) T_2 \\ Y_1^{T_2} + Y_0^{T_2} &= T_2 (1 - \mathcal{O}(1/\omega)). \end{aligned}$$

Before we move to the proof of this proposition, let us make some simple but useful observations. First, note that only a negligible fraction of the nodes have opinion that we do not control. The proof is deferred to the full version.

► **Lemma 9.** *Suppose that  $p = p(n) \ll 1$  and  $p \gg 1/n$ . Let  $\omega = \omega(n) \ll \min\{pn, (1/p)^{1/2}\}$  be any function that tends to infinity as  $n \rightarrow \infty$ . Set  $T_2 = T_2(n)$  such that  $\omega/p \leq T_2 \leq n/\omega$ . Then, a.a.s., for any  $t$  such that  $T_1 \leq t \leq T_2$ ,  $Z_1^t \leq 2t/\omega$ .*

Let us fix  $k \in \mathbb{N}$  and consider random variable  $X_k \in \text{Bin}(k, 1/2 + \delta/2)$ . We will need to understand the following sequence of constants (the connection to our problem will become clear soon):

$$q_k := \mathbb{P}(X_k > k/2) + \mathbb{P}(X_k = k/2) \cdot (1/2 + \delta). \quad (10)$$

Clearly,  $q_0 = 1/2 + \delta$  and  $q_1 = 1/2 + \delta/2$ . For any other value of  $k \geq 2$ ,  $q_k \geq 1/2 + 51\delta/100$  as we show in the next technical lemma. The proof can be found in the full version.

► **Lemma 10.** *Fix  $k \in \mathbb{N}$  such that  $k \geq 2$ , and  $\delta \in (0, 1/10]$ . Then,*

$$q_k \geq \frac{1}{2} + \frac{51}{100}\delta.$$

Now, we are ready to go back to analyzing the behaviour of the process during the second phase.

**Proof of Proposition 8.** Our goal is to show that a.a.s. the following inequalities hold for any  $t$  such that  $T_1 \leq t \leq T_2$ :

$$\frac{Z_1^t}{Z_0^t + Z_1^t} \geq \frac{1/2 + \delta/2}{1/2 - \delta/2} \quad (11)$$

$$Z_1^t \leq 2t/\omega. \quad (12)$$

Formally, we define the stopping time  $S$  to be the minimum value of  $t \geq T_1$  such that either (11) fails, (12) fails or  $t = T_2$ . (A stopping time is any random variable  $S$  with values in  $\{T_1, T_1 + 1, \dots, T_2\}$  such that, for any time  $\hat{t}$ , it is determined whether  $S = \hat{t}$  from knowledge of the process up to and including time  $\hat{t}$ .)

Property (12) is trivially satisfied at the beginning of the second phase as  $Z_1^{T_1} = 0$ . By Proposition 6, since we aim for a statement that holds a.a.s., we may assume that (11) is satisfied at the beginning of the second phase. In fact,

$$\frac{Z_1^{T_1}}{Z_0^{T_1} + Z_1^{T_1}} = \frac{Y_1^{T_1}}{Y_0^{T_1} + 0} \geq \frac{1/2 + 101\delta/200}{1/2 - 101\delta/200} \geq \frac{1/2 + \delta/2}{1/2 - \delta/2}.$$

It will be convenient to define  $Z^t = Z_1^t + Z_0^t + Z_2^t$ ; that is,  $Z^t$  is the number of nodes that announced their opinions by time  $t$ . If (12) is satisfied, then only a negligible fraction of nodes were selected more than once and we get that  $Z^t = t(1 - \mathcal{O}(1/\omega)) \sim t$ .

Let us first show that if (11) and (12) are satisfied at time  $t$  and the node selected at time  $t + 1$  was not selected before (that is,  $D^t(v^{t+1}) = C^t(v^{t+1}) = \perp$ ), then the probability that  $v^{t+1}$  announces an auxiliary opinion 1 is at least  $1/2 + 101/200\delta$ .

We first expose edges from  $v^{t+1}$  to the set  $\{v : D^t(v) \neq \perp\}$  (see Property 5) and let us define  $p_k$  to be the probability that  $v^{t+1}$  has precisely  $k$  neighbours in that set. In particular, we have

$$\begin{aligned} p_1 &= Z^t p(1-p)^{Z^t-1} = \lambda(1-p)^{\lambda/p-1} \\ &\leq \lambda e^{-\lambda}/(1-p) \\ &\leq 1/e + o(1) < 1/2, \end{aligned} \tag{13}$$

where  $\lambda = pZ^t = pt(1 - \mathcal{O}(1/\omega))$  and the second inequality follows because  $xe^{-x} \leq e^{-1}$  and  $p = o(1)$ .

Now, condition on  $v^t$  having exactly  $k$  neighbours that already announced their opinion. Note that we did not expose the neighbours yet (only the number of them) so neighbours form a random set of cardinality  $k$  from the set  $\{v : D^{t-1}(v) \neq \perp\}$ . Let  $r_k$  to be the probability that  $v^t$  announces auxiliary opinion 1 in this conditional probability space. It happens if more than  $k/2$  neighbours of  $v^t$  have  $D^{t-1}(v) = 1$ . Moreover, if exactly  $k/2$  neighbours have this property, then  $v^t$  announces opinion 1 with probability  $1/2 + \delta$ , which is the probability that its private belief is 1. Since (11) holds,  $r_k$  can be lower bounded by  $q_k$  which we defined in (10). It follows that the probability that  $v^t$  announces 1 is asymptotic to

$$\begin{aligned} \sum_{k \geq 0} r_k \cdot p_k &\geq \sum_{k \geq 0} q_k \cdot p_k = q_1 p_1 + \sum_{k \geq 0, k \neq 1} q_k \cdot p_k \\ &\geq \left(\frac{1}{2} + \frac{\delta}{2}\right) p_1 + \left(\frac{1}{2} + \frac{51}{100}\delta\right) (1 - p_1) \\ &= \left(\frac{1}{2} + \frac{51}{100}\delta\right) - p_1 \left(\frac{1}{100}\delta\right) \\ &\geq \frac{1}{2} + \frac{101}{200}\delta, \end{aligned} \tag{14}$$

where the second inequality follows from Lemma 10 and the last one from (13).

Let  $s$  be the number of rounds  $t$  in the second phase in which  $v^t$  was not selected before, i.e.,  $D^{t-1}(v^t) = \perp$ , and let  $t_1, t_2, \dots, t_s$  denote such round. Clearly,  $s \leq T_2 - T_1 = T_2(1 - \mathcal{O}(1/\omega))$  but, in fact, a.a.s. we have  $s = T_2(1 - \mathcal{O}(1/\omega))$  by Lemma 9. For  $i \in [s]$ , let  $L_i$  be the indicator random variable for the event that  $v^{t_i}$  announced an auxiliary opinion 1, that is,  $L_i = Z_1^{t_i} - Z_1^{t_i-1}$ . If both (11) and (12) hold at time  $t_i - 1$ , then  $\mathbb{P}(L_i = 1) \geq 1/2 + 101\delta/200$  but, of course, we cannot condition on these two properties to hold. Instead, we will use a small trick and consider an auxiliary sequence of random variables after the stopping time  $S$  when one of the properties fails.

Fix  $\hat{p} = 1/2 + 101\delta/200$  and let  $M_1, \dots, M_s$  be a sequence of independent Bernoulli variables with parameter  $\hat{p}$ . For each  $i \in [s]$ , we define  $L'_i = L_i$  if both (11) and (12) hold at times  $t < t_i$  and otherwise  $L'_i = M_i$ . That is, the process “stops” at our stopping time  $S$  which, in our context, means that it simply follows part of the sequence  $(M_i)_{i=1}^s$  (namely,  $(M_i)_{i=S+1}^s$ ) from that point on, ignoring the behaviour of the original process. Thus, defining  $L'_{\leq j} = \sum_{i=1}^j L'_i$  and  $M_{\leq j} = \sum_{i=1}^j M_i$ , (14) implies that one can couple  $L'_{\leq j}$  and  $M_{\leq j}$  such that  $L'_{\leq j} \geq M_{\leq j}$  for all  $j \in [s]$ .

Note that  $\mathbb{E}[M_{\leq j}] = \hat{p}j = (1/2 + 101\delta/200)j$  for any  $j \in [s]$ . It follows from Chernoff’s bound (4),

$$\begin{aligned}
 & \mathbb{P}\left(\exists_{1 \leq j \leq s} \mathbb{E}[M_{\leq j}] - M_{\leq j} \geq \frac{\delta}{400}(T_1 + j)\right) \\
 & \leq \sum_{a \geq 1} \mathbb{P}\left(\max_{(2^{a-1}-1)T_1 < j \leq (2^a-1)T_1} (\mathbb{E}[M_{\leq j}] - M_{\leq j}) \geq 2^{a-1} \frac{\delta}{400} T_1\right) \\
 & \leq \sum_{a \geq 1} \exp(-\Theta(2^a T_1)) = \exp(-\Theta(T_1)) = o(1),
 \end{aligned}$$

since  $T_1 = \Theta(1/p) \rightarrow \infty$ . In other words, a.a.s. for any  $j \in [s]$ ,

$$L'_{\leq j} \geq M_{\leq j} \geq \left(\frac{1}{2} + \frac{101}{200}\delta\right)j - \frac{\delta}{400}(T_1 + j).$$

Since  $L_{\leq j} = L'_{\leq j}$  for any  $j \in [s]$  such that  $t_j < S$ , and  $Z_1^t$  can decrease by at most one in a single round, a.a.s.

$$\begin{aligned}
 Z_1^S & \geq Z_1^{S-1} - 1 \\
 & \geq Z_1^{T_1} + L_{\leq S-T_1-\mathcal{O}(S/\omega)} - \mathcal{O}(S/\omega) \\
 & \geq \left(\frac{1}{2} + \frac{101}{200}\delta\right)T_1 + \left(\frac{1}{2} + \frac{101}{200}\delta\right)(S - T_1 - \mathcal{O}(S/\omega)) - \frac{\delta}{400}(S - \mathcal{O}(S/\omega)) - \mathcal{O}(S/\omega) \\
 & \geq \left(\frac{1}{2} + \frac{201}{400}\delta\right)S - \mathcal{O}(S/\omega) \\
 & \geq \left(\frac{1}{2} + \frac{\delta}{2}\right)S,
 \end{aligned}$$

implying that (11) holds at time  $S$ . Indeed, there were  $Z_1^{T_1}$  nodes with auxiliary opinion 1 at the beginning of the second phase. By Lemma 9,  $S - T_1 - \mathcal{O}(S/\omega)$  nodes were selected for the first time before the stopping time and  $L_{\leq S-T_1-\mathcal{O}(S/\omega)}$  of them announced 1 at that time. Finally, at most  $\mathcal{O}(S/\omega)$  nodes that already announced their opinion were selected again. It implies that a.a.s. the process does not “stop” because of (11) failing. By Lemma 9, a.a.s. it also does not stop because of (12). Hence, a.a.s.  $S = T_2$  and the proof of the proposition is finished.  $\blacktriangleleft$

### 3.3 Not Very Sparse Random Graphs

In this subsection, we provide a relatively easy argument that works for random graphs with  $pn \gg \log n$ . In particular, we show that a.a.s. after round  $T_2$  but before round  $T_3 = n/\sqrt{\omega}$  all nodes that are selected for the first time announce 1. Moreover, after round  $T_3$  every node selected announces 1 a.a.s.

**Proof of Theorem 1.** Let  $\omega = \omega(n) \ll \min\{(pn/\log n)^{1/2}, pn, (1/p)^{1/2}\}$  be any function that tends to infinity as  $n \rightarrow \infty$ . In particular,  $pn \geq \omega^2 \log n$ . Fix  $T_2 = T_2(n) = n/\omega$ . It follows from Proposition 8 that a.a.s. at the end of the second phase, there are  $Y_1^{T_2} \geq (1/2 + \delta/2)T_2$  nodes that announced opinion 1, and so  $Y_0^{T_2} \leq (1/2 - \delta/2)T_2$  nodes announced opinion 0; moreover,  $Y_1^{T_2} + Y_0^{T_2} = T_2(1 + \mathcal{O}(1/\omega))$ .

Let  $V_i = \{v : C^t(v) = i\}$  be the set of nodes with opinion  $i \in \{0, 1\}$  at time  $T_2$ . Note that, by Property 5, we may assume that only edges within  $V_0 \cup V_1$  are exposed at that stage of the process. We will first show that a.a.s. all nodes  $v \notin V_0 \cup V_1$  have substantially more neighbours in  $V_1$  than in  $V_0$ . Indeed, this is a simple consequence of the Chernoff bounds (1) and (2): for any  $v \notin V_0 \cup V_1$ :

$$\begin{aligned}
& \mathbb{P}\left(|N(v) \cap V_1| \leq |N(v) \cap V_0| + \delta T_2 p / 2\right) \\
& \leq \Pr\left(|N(v) \cap V_1| \leq (1/2 - \delta/4)T_2 p + \delta T_2 p / 2 \text{ or } |N(v) \cap V_0| \geq (1/2 - \delta/4)T_2 p\right) \\
& \leq \mathbb{P}\left(|N(v) \cap V_1| \leq (1/2 + \delta/4)T_2 p\right) + \mathbb{P}\left(|N(v) \cap V_0| \geq (1/2 - \delta/4)T_2 p\right) \\
& = \mathbb{P}\left(\text{Bin}(|V_1|, p) \leq (1/2 + \delta/4)T_2 p\right) + \mathbb{P}\left(\text{Bin}(|V_0|, p) \geq (1/2 - \delta/4)T_2 p\right) \\
& \leq 2 \exp\left(-\Theta(T_2 p)\right) \\
& = 2 \exp\left(-\Omega\left(\frac{n}{\omega} \cdot \frac{\omega^2 \log n}{n}\right)\right) \\
& = \mathcal{O}(1/n^2),
\end{aligned}$$

where the first inequality follows simply by observing that  $|N(v) \cap V_1| > c + \delta T_2 p / 2$  and  $|N(v) \cap V_0| < c$  implies  $|N(v) \cap V_1| > |N(v) \cap V_0| + \delta T_2 p / 2$ . The final inequality follows since  $\mathbb{E}[\text{Bin}(|V_1|, p)] \geq (1/2 + \delta/2)T_2 p$  and  $\mathbb{E}[\text{Bin}(|V_0|, p)] \leq (1/2 - \delta/2)T_2 p$ . The desired property holds by the union bound over all nodes  $v \notin V_0 \cup V_1$ .

Fix  $T_3 = T_3(n) = n/\sqrt{\omega}$ . The third phase will last till time  $T_3$ . Let  $V'_1 \subseteq V_1$  be the set of nodes from  $V_1$  that were selected during the third phase. Note that each node from  $V_1$  is selected during the third phase with probability at most  $(T_3 - T_2)/n \leq 1/\sqrt{\omega}$ . Hence,  $\mathbb{E}[|V'_1|] \leq |V_1|/\sqrt{\omega}$  and so a.a.s.  $|V'_1| \leq |V_1|/\omega^{1/3}$  by Markov's inequality. A simple but important observation is that  $V'_1$  is determined exclusively by the selection process (coupon collector process); in particular, it does not depend on the random graph nor the opinion dynamics. Hence, we can use Chernoff's bound again to show that a.a.s. all nodes  $v \notin V_0 \cup V_1$  have very few neighbours in  $V'_1$ . Indeed, note that for any  $v \notin V_0 \cup V_1$ , the number of neighbours of  $n$  in  $V'_1$  can be stochastically upper bounded by  $\text{Bin}(|V_1|/\omega^{1/3}, p)$  with expectation  $|V_1|p/\omega^{1/3} = \Theta(np/\omega^{4/3}) = \Omega(n^{2/3} \log n) \gg \log n$ . Hence,  $|N(v) \cap V'_1| = \mathcal{O}(|V_1|p/\omega^{1/3}) = \mathcal{O}(T_2 p/\omega^{1/3}) = o(T_2 p)$  with probability  $1 - \mathcal{O}(1/n^2)$ , and so a.a.s. all nodes  $v \notin V_0 \cup V_1$  satisfy this property.

Combining the two properties together, we get that a.a.s. for all nodes  $v \notin V_0 \cup V_1$  we have

$$|N(v) \cap (V_1 \setminus V'_1)| > |N(v) \cap (V_0 \cup V'_1)|. \quad (15)$$

Let  $W_1$  be the set of nodes outside of  $V_0 \cup V_1$  that were selected during the third phase (possibly multiple times). If property (15) is satisfied, then (deterministically) all nodes in  $W_1$  announce 1 in this phase. Indeed, even if all nodes from  $V'_1$  changed their opinion to 0 in the meantime, nodes in  $V_1$  still have majority of their neighbours with opinion 1.

Let us summarize the situation at the beginning of the fourth (and the last) phase. Recall that  $W_1$  consists of nodes that were selected for the first time during the third phase. Let  $W_0 = V_0 \cup V_1$  be the set of nodes that were selected before the third phase (that is, during the first or the second phase). A.a.s. nodes in  $W_1$  have opinion 1 and  $|W_1| = (T_3 - T_2) + \mathcal{O}(T_3^2/n) \sim T_3$ . We may assume that nodes in  $W_0$  have opinion 0 and a.a.s.  $|W_0| = T_2(1 + \mathcal{O}(1/\omega)) \sim T_2 = o(T_3)$ . Again, it is important to notice that  $W_1$  and  $W_0$  are determined exclusively by the selection process. ( $V_1$  and  $V_0$  do not possess this property and that was the main reason we needed to consider the third phase.) We may then use Chernoff's bound again, on the number of neighbours in  $W_1$  and  $W_0$  of any given node, to show that a.a.s. all nodes (not only outside of  $W_1 \cup W_0$ !) have more neighbours in  $W_1$  than in  $W_0$ . It means that every node that is selected during this last phase announces opinion 1.

Since a.a.s. every node is selected at least once during the next  $n(\log n + \omega'/2)$  rounds, the process is over after at most that many rounds with everyone converging to opinion 1. Hence, a.a.s. the entire process takes at most  $T_3 + n(\log n + \omega'/2) \leq n(\log n + \omega')$  rounds. In fact, the expected number of nodes that were selected before the last phase but were not selected in the first  $T'_4 = n(\log n - \log \omega/4)$  rounds of the last phase is equal to

$$T_3 \left(1 - \frac{1}{n}\right)^{T'_4} \leq \frac{n}{\sqrt{\omega}} \exp(-\log n + \frac{1}{4} \log \omega) = \omega^{-1/4} = o(1),$$

and so a.a.s. all nodes selected before the last phase are selected again during the first  $T'_4$  rounds of the last phase. On the other hand, a.a.s. there are still some nodes not selected at all after  $T_3 + T'_4$  rounds. Indeed, this follows immediately from the well studied coupon collector concentration bound for  $\hat{T}$ :  $\mathbb{P}(\hat{T} < n \log n - cn) < e^{-c}$ . The conclusion is that a.a.s. all nodes are selected at least once between round  $T_3$  and  $\hat{T}$ , and the proof is finished.  $\blacktriangleleft$

### 3.4 Very Sparse Random Graphs

In this subsection, we investigate random graphs that are close to the threshold for connectivity but are still connected, that is, we assume that  $pn \leq \omega \log n$  and  $pn \geq \log n + \omega$  for some  $\omega = \omega(n) \rightarrow \infty$  as  $n \rightarrow \infty$ .

First, we will show that at time  $T_3 = T_3(n) = 2n \log n$ , every node announced its opinion at least once, and only at most  $n\omega/\log n = o(n)$  nodes have opinion 0. The proof is deferred to the full version.

**► Proposition 11.** *Let  $\omega = \omega(n) = o(\log n)$  be any function that tends to infinity (sufficiently slowly) as  $n \rightarrow \infty$ . Suppose that  $p = p(n) \leq \omega \log n/n$  and  $p \geq (\log n + \omega)/n$ . Set  $T_3 = T_3(n) = 2n \log n$  and  $s = s(n) = n\omega/\log n$ . Then, a.a.s. all nodes announced their opinion at time  $T_3$ , and at most  $s$  of them have opinion 0.*

We will call a node  $v$  to be of *small degree*, if its degree is at most  $k = 5 \log n / (\log \log n)^{1/2}$ . Nodes of degree larger than  $k$  will be called of *large degree*. Before we continue investigating the process, we need to show a well-known fact that small degree nodes are not too close to each other.

**► Lemma 12.** *Let  $\omega = \omega(n) = o(\log n)$  be any function that tends to infinity (sufficiently slowly) as  $n \rightarrow \infty$ . Suppose that  $p = p(n) \leq \omega \log n/n$  and  $p \geq (\log n + \omega)/n$ . Then, the following property holds a.a.s. in  $\mathcal{G}(n, p)$ : any two small degree nodes are at distance at least 2 from each other.*

**Proof.** Since  $np > \log n$  and  $k = o(\log n)$ ,  $\binom{n}{i} p^i$  is an increasing sequence for  $0 \leq i \leq k$  and hence we have

$$\mathbb{P}(\deg(v) \leq k) \leq \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i} \leq (k+1) \binom{n}{k} p^k (1-p)^{n-k}.$$

We obtain the following upper bound on the probability that a node  $v$  has small degree:

$$\begin{aligned} \mathbb{P}(\deg(v) \leq k) &\leq (k+1) \binom{n}{k} p^k (1-p)^{n-k} \\ &\leq (k+1) \left(\frac{en}{k} \cdot \frac{\omega \log n}{n}\right)^k \exp(-pn + pk) \\ &\leq (k+1) \left(\omega (\log \log n)^{1/2}\right)^k \exp(-\log n - \omega + o(1)) \\ &\leq (k+1) \exp\left(\frac{5 \log n}{(\log \log n)^{1/2}} \cdot \log \log \log n - \log n\right) \\ &= n^{-1+o(1)}. \end{aligned}$$



Hence, we expect  $n^{o(1)}$  small degree nodes and so a.a.s. we have only  $n^{o(1)}$  of them. More importantly, using similar computations one can show that the expected number of small degree nodes that are adjacent to each other is equal to

$$\binom{n}{2} \cdot p \cdot \left(n^{-1+o(1)}\right)^2 = n^{-1+o(1)} = o(1).$$

Similarly, the expected number of small degree nodes that are at distance two from each other is equal to

$$\binom{n}{2} \cdot n \cdot p^2 \cdot \left(n^{-1+o(1)}\right)^2 = n^{-1+o(1)} = o(1).$$

Hence, a.a.s. any two nodes of small degree are at distance at least two from each other, and the proof of the lemma is finished.  $\blacktriangleleft$

Our next observation is that the number of large degree nodes that have opinion 0 is decreasing. The proof is deferred to the full version.

► **Proposition 13.** *Let  $\omega = \omega(n) = o(\log n)$  be any function that tends to infinity (sufficiently slowly) as  $n \rightarrow \infty$ . Suppose that  $p = p(n) \leq \omega \log n/n$  and  $p \geq (\log n + \omega)/n$ . Then, the following property holds a.a.s. for all phases.*

*Suppose that at the beginning of a phase,  $s = (n\omega/\log n) \cdot (\log \log n)^{-(i-1)/4}$  large degree nodes have opinion 0 for some  $i \in \mathbb{N}$ . Then, after  $2n \log n$  rounds all nodes announced their opinion at least once more, and at most  $u = s/(\log \log n)^{1/4} = (n\omega/\log n) \cdot (\log \log n)^{-i/4}$  large degree nodes have opinion 0.*

Finally, we are ready to show that all nodes eventually converge to opinion 1.

**Proof of Theorem 2.** The proof is an easy consequence of Propositions 11, 13, and Lemma 12. Indeed, a.a.s. at time  $T_3 = T_3(n) = 2n \log n$ , all but at most  $s = s(n) = n\omega/\log n$  nodes have opinion 1 (Proposition 11). Most of them are of large degree but some of them may be of small degree. By Proposition 13, the number of large degree nodes that have opinion 0 decreases: a.a.s. at time  $2n \log n \cdot \mathcal{O}(\log n/\log \log n) = \mathcal{O}(n(\log n)^2/(\log \log n))$  no large degree node has opinion 0. There could possibly be still some nodes of small degree that have opinion 0 but everyone converges to opinion 1 after additional  $\mathcal{O}(n \log n)$  rounds. Indeed, every node is selected at least once during that time period a.a.s. Large degree nodes have many neighbours but at most one neighbour of small degree (Lemma 12). So they will not change their opinion and stay with opinion 1. On the other hand, by the same lemma, no small degree node has a neighbour of small degree. Hence, such nodes will switch to opinion 1 once they are selected again. This finishes the proof of the theorem.  $\blacktriangleleft$

## 4 Dense Random Graphs

In this section, we prove that for dense graphs (that is, when  $p \in (0, 1]$  is a constant) it is not true that all nodes converge to the correct opinion a.a.s. On the contrary, there maybe an information cascade where all the nodes converge to the wrong opinion with constant probability.

**Proof of Theorem 3.** Fix any  $p \in (0, 1)$ . We will consider the easy case  $p = 1$  at the end.

Trivially, the first node announces its private belief, that is, it announces opinion 1 with probability  $1/2 + \delta$ ; otherwise, it announces 0. Since nodes are selected by the process (“coupon collector”) independently of the graph, we may postpone exposing edges of the

random graph till the first time a node is selected. Each time this happens, we expose edges from  $v^t$  to all nodes that already announced their opinion. If every single time at least one edge is present, then all nodes are going to announce the opinion of the very first node. It follows that

$$p_1 \geq (1/2 + \delta) \prod_{i=1}^n (1 - (1-p)^i).$$

It is easy to see that for any  $x \in [0, 1-p]$ ,

$$f(x) = 1 - x \geq \exp\left(-\frac{\log(1/p)}{1-p}x\right) = g(x).$$

(Note that  $f(0) = g(0)$ ,  $f(1-p) = g(1-p)$ , and  $g(x)$  is convex.) Hence,

$$\begin{aligned} p_1 &\geq (1/2 + \delta) \exp\left(-\frac{\log(1/p)}{1-p} \sum_{i=1}^n (1-p)^i\right) \\ &\geq (1/2 + \delta) \exp\left(-\log(1/p) \sum_{i=0}^{\infty} (1-p)^i\right) \\ &= (1/2 + \delta) \exp\left(-\log(1/p)(1/p)\right). \end{aligned}$$

The same argument works for  $p_0$  with the only difference that the probability of the first node announcing 1 ( $1/2 + \delta$ ) needs to be replaced with the probability of announcing 0 ( $1/2 - \delta$ ).

Finally, note that if  $p = 1$ , then the graph is (deterministically) the complete graph and (again, deterministically) all nodes are going to adopt the opinion of the very first node. Thus, we immediately get  $p_1 = 1/2 + \delta$  and  $p_0 = 1/2 - \delta$  (which matches the general formula that works for  $p \in (0, 1]$ ). This finishes the proof of the theorem. ◀

---

## References

- 1 Mohammed Amin Abdullah and Moez Draief. Global majority consensus by local majority polling on graphs of a given degree sequence. *Discrete Applied Mathematics*, 180:1–10, 2015.
- 2 Daron Acemoglu, Munther A. Dahleh, Ilan Lobel, and Asuman Ozdaglar. Bayesian learning in social networks. *Review of Economic Studies*, 78(4):1201–1236, 2011.
- 3 Aris Anagnostopoulos, Luca Becchetti, Emilio Cruciani, Francesco Pasquale, and Sara Rizzo. Biased opinion dynamics: when the devil is in the details. *Information Sciences*, 593:49–63, 2022.
- 4 Maryam Bahrani, Nicole Immorlica, Divyarthi Mohan, and S Matthew Weinberg. Asynchronous majority dynamics in preferential attachment trees. *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, 2020.
- 5 Abhijit Banerjee, Emily Breza, Arun G Chandrasekhar, and Markus Mobius. Naive learning with uninformed agents. *American Economic Review*, 111(11):3540–3574, 2021.
- 6 Abhijit Banerjee and Drew Fudenberg. Word-of-mouth learning. *Games and Economic Behavior*, 46(1):1–22, January 2004. URL: <http://ideas.repec.org/a/eee/gamebe/v46y2004i1p1-22.html>.
- 7 Abhijit V. Banerjee. A simple model of herd behavior. *The Quarterly Journal of Economics*, 107(3):797–817, 1992.
- 8 Luca Becchetti, Andrea Clementi, and Emanuele Natale. Consensus dynamics: An overview. *SIGACT News*, 51(1):58–104, March 2020. doi:10.1145/3388392.3388403.
- 9 Luca Becchetti, Andrea E.F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *ACM Symposium on Discrete Algorithms (SODA)*, 2016.

- 10 Itai Benjamini, Siu-On Chan, Ryan O'Donnell, Omer Tamuz, and Li-Yang Tan. Convergence, unanimity and disagreement in majority dynamics on unimodular graphs and random graphs. *Stochastic Processes and their Applications*, 126(9):2719–2733, 2016.
- 11 Sushil Bikhchandani, David Hirshleifer, Omer Tamuz, and Ivo Welch. Information cascades and social learning. Technical report, National Bureau of Economic Research, 2021.
- 12 Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change in informational cascades. *Journal of Political Economy*, 100(5):992–1026, October 1992.
- 13 Béla Bollobás. *Random graphs*. Cambridge University Press, 2001.
- 14 Boğaçhan Çelen and Shachar Kariv. Observational learning under imperfect information. *Games and Economic behavior*, 47(1):72–86, 2004.
- 15 Debsoumya Chakraborti, Jeong Han Kim, Joonkyung Lee, and Tuan Tran. Majority dynamics on sparse random graphs. *Random Structures & Algorithms*, 63(1):171–191, 2023.
- 16 Fan RK Chung and Linyuan Lu. *Complex graphs and networks*. American Mathematical Soc., 2006.
- 17 Peter Clifford and Aidan Sudbury. A model for spatial conflict. *Biometrika*, 60(3):581–588, 1973. URL: <http://www.jstor.org/stable/2335008>.
- 18 Emilio Cruciani, Hlafo Alfie Mimun, Matteo Quattropiani, and Sara Rizzo. Phase transitions of the k-majority dynamics in a biased communication model. In *Proceedings of the 22nd International Conference on Distributed Computing and Networking*, pages 146–155, 2021.
- 19 Morris H. DeGroot. Reaching a consensus. *Review of Economic Studies*, 69(345):118–121, 1974.
- 20 Michal Feldman, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. Reaching consensus via non-bayesian asynchronous learning in social networks. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014*, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.192.
- 21 Nikolaos Fountoulakis, Mihyun Kang, and Tamás Makai. Resolution of a conjecture on majority dynamics: Rapid stabilization in dense random graphs. *Random Structures & Algorithms*, 57(4):1134–1156, 2020.
- 22 Douglas Gale and Shachar Kariv. Bayesian learning in social networks. *Games and Economic Behavior*, 45(2):329–346, 2003. Special Issue in Honor of Robert W. Rosenthal. doi:10.1016/S0899-8256(03)00144-1.
- 23 Bernd Gärtner and Ahad N Zehmakan. Majority model on random regular graphs. In *LATIN 2018: Theoretical Informatics: 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings 13*, pages 572–583. Springer, 2018.
- 24 Mohsen Ghaffari and Johannes Lengler. Nearly-tight analysis for 2-choice and 3-majority consensus dynamics. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2018.
- 25 Mohsen Ghaffari and Merav Parter. A polylogarithmic gossip algorithm for plurality consensus. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2016.
- 26 Benjamin Golub and Matthew O. Jackson. Naïve learning in social networks and the wisdom of crowds. *American Economic Journal: Microeconomics*, 2(1):112–149, 2010.
- 27 Richard A. Holley and Thomas M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *The Annals of Probability*, 3(4):643–663, 1975. URL: <http://www.jstor.org/stable/2959329>.
- 28 C Douglas Howard. Zero-temperature ising spin dynamics on the homogeneous tree of degree three. *Journal of applied probability*, 37(3):736–747, 2000.
- 29 Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random graphs*, volume 45. John Wiley & Sons, 2011.
- 30 Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Network Science*, 9(2):153–178, 2021.

- 31 Y. Kanoria and O. Tamuz. Tractable bayesian social learning on trees. *IEEE Journal on Selected Areas in Communications*, 31(4):756–765, 2013.
- 32 Yashodhan Kanoria, Andrea Montanari, et al. Majority dynamics on trees and the dynamic cavity method. *The Annals of Applied Probability*, 21(5):1694–1748, 2011.
- 33 Michal Karoński and Alan Frieze. *Introduction to Random Graphs*. Cambridge University Press, 2016.
- 34 Marcos Kiwi, Lyuben Lichev, Dieter Mitsche, and Paweł Prałat. Label propagation on binomial random graphs. *arXiv preprint*, 2023. [arXiv:2302.03569](https://arxiv.org/abs/2302.03569).
- 35 S. Matwin, A. Milios, P. Prałat, A. Soares, and F. Théberge. *Generative Methods for Social Media Analysis*. SpringerBriefs in Computer Science. Springer Nature Switzerland, 2023. URL: <https://books.google.ca/books?id=wPbJEAAAQBAJ>.
- 36 Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, pages 195–248. Springer, 1998.
- 37 Markus Mobius and Tanya Rosenblat. Social learning in economics. *Annual Review of Economics*, 6(1):827–847, 2014. doi:10.1146/annurev-economics-120213-012609.
- 38 E. Mossel, N. Olsman, and O. Tamuz. Efficient bayesian learning in social networks with gaussian estimators. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016.
- 39 Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2013.
- 40 Elchanan Mossel, Allan Sly, and Omer Tamuz. Asymptotic learning on bayesian social networks. *Probability Theory and Related Fields*, 2014.
- 41 Manuel Mueller-Frank. A general framework for rational learning in social networks. *Theoretical Economics*, 8(1):1–40, 2013.
- 42 Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Informational externalities and emergence of consensus. *Games and Economic Behavior*, 66(2):979–994, 2009.
- 43 Lones Smith and Peter Sorensen. Pathological outcomes of observational learning. *Econometrica*, 68(2):371–398, March 2000. URL: <http://ideas.repec.org/a/econ/emetrp/v68y2000i2p371-398.html>.
- 44 Omer Tamuz and Ran Tessler. Majority dynamics and the retention of information. In *Working paper*, 2013.
- 45 Linh Tran and Van Vu. Reaching a consensus on random networks: The power of few. *arXiv preprint*, 2019. [arXiv:1911.10279](https://arxiv.org/abs/1911.10279).
- 46 Linh Tran and Van Vu. The “power of few” phenomenon: The sparse case. *arXiv preprint*, 2023. [arXiv:2302.05605](https://arxiv.org/abs/2302.05605).
- 47 Ahad N Zehmakan. Opinion forming in Erdős–Rényi random graph and expanders. *Discrete Applied Mathematics*, 277:280–290, 2020.

# Bipartizing (Pseudo-)Disk Graphs: Approximation with a Ratio Better than 3

Daniel Lokshtanov  

University of California, Santa Barbara, USA

Fahad Panolan  

University of Leeds, UK

Saket Saurabh  

Institute of Mathematical Sciences, India

Jie Xue  

New York University Shanghai, China

Meirav Zehavi  

Ben-Gurion University, Israel

---

## Abstract

In a disk graph, every vertex corresponds to a disk in  $\mathbb{R}^2$  and two vertices are connected by an edge whenever the two corresponding disks intersect. Disk graphs form an important class of geometric intersection graphs, which generalizes both planar graphs and unit-disk graphs. We study a fundamental optimization problem in algorithmic graph theory, BIPARTIZATION (also known as ODD CYCLE TRANSVERSAL), on the class of disk graphs. The goal of BIPARTIZATION is to delete a minimum number of vertices from the input graph such that the resulting graph is bipartite. A folklore (polynomial-time) 3-approximation algorithm for BIPARTIZATION on disk graphs follows from the classical framework of Goemans and Williamson [Combinatorica'98] for cycle-hitting problems. For over two decades, this result has remained the best known approximation for the problem (in fact, even for BIPARTIZATION on unit-disk graphs). In this paper, we achieve the first improvement upon this result, by giving a  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on disk graphs, for some constant  $\alpha > 0$ . Our algorithm directly generalizes to the broader class of *pseudo-disk* graphs. Furthermore, our algorithm is *robust* in the sense that it does not require a geometric realization of the input graph to be given.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** bipartization, geometric intersection graphs, approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.6

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.09356>

## 1 Introduction

Disk graphs refer to intersection graphs of disks in the plane  $\mathbb{R}^2$ . Formally, in a disk graph, every vertex corresponds to a disk in  $\mathbb{R}^2$  and two vertices are connected by an edge whenever the two corresponding disks intersect. As a rather general class of geometric intersection graphs, disk graphs simultaneously generalize two important graph classes, unit-disk graphs and planar graphs, both of which have been extensively studied over decades. Many central problems in algorithmic graph theory have been considered on disk graphs, including VERTEX COVER [8, 28, 41], INDEPENDENT SET [8], MAXIMUM CLIQUE [5], FEEDBACK VERTEX SET [26, 28], DOMINATING SET [13], etc.



© Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 6; pp. 6:1–6:14



Leibniz International Proceedings in Informatics

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

In this paper, we investigate a fundamental optimization problem on the class of disk graphs, called BIPARTIZATION. In this problem, the input is a graph  $G$  and the goal is to delete a smallest number of vertices from  $G$  such that the resulting graph is bipartite. There has been a long line of work studying BIPARTIZATION (e.g., see [10, 20, 34, 36, 37] and citations therein). Observe that the edge counterpart of the BIPARTIZATION problem, EDGE BIPARTIZATION, where we need to find fewest edges whose deletion results in a bipartite graph, is equivalent to the classical MAXIMUM CUT problem (which has been studied for over five decades [14, 39]). It is known that EDGE BIPARTIZATION (and MAXIMUM CUT) reduces to BIPARTIZATION [42], and both problems find applications in computational biology [42, 38], VLSI chip design [18], genome sequence assembly [32], and more. BIPARTIZATION is of particular interest also due to the following observation: a graph is bipartite if and only if it does not contain any odd cycle. As such, it can be formulated as hitting all odd cycles in the graph (using fewest vertices). For this reason, the BIPARTIZATION problem is also known as ODD CYCLE TRANSVERSAL, and belongs to the family of *cycle-hitting* problems, one of the most well-studied topics in algorithmic graph theory. Besides BIPARTIZATION, other important cycle-hitting problems that have been studied on disk graphs include FEEDBACK VERTEX SET, TRIANGLE HITTING, SHORT CYCLE HITTING, etc.

There is no surprise that BIPARTIZATION is NP-complete. In fact, it is NP-complete even on graphs of maximum degree 3 and planar graphs of maximum degree 4 [6]. As such, the study of the problem is mainly in the context of approximation algorithms and parameterized algorithms. On the parameterized front, it was known that BIPARTIZATION can be solved in  $2^{O(k)} \cdot n^{O(1)}$  time where  $k$  is the solution size [20, 25, 35]. On planar graphs and unit-disk graphs, there exist improved algorithms with running time  $k^{O(\sqrt{k})} \cdot n^{O(1)}$  [2, 3, 29, 31], which are almost tight assuming the Exponential-Time Hypothesis (ETH). On disk graphs, it was not known whether BIPARTIZATION admits a subexponential-time parameterized algorithm, until very recently Lokshantov et al. gave a  $k^{O(k^{27/28})} \cdot n^{O(1)}$ -time algorithm [26].

From the perspective of approximation algorithms (which is the focus of this paper), BIPARTIZATION is one of the trickiest problems in the sense that no polynomial-time approximation scheme (PTAS) was known on any (nontrivial) graph classes, but at the same time no inapproximability results was known except for general graphs. On general graphs, BIPARTIZATION cannot admit any (polynomial-time) constant-approximation algorithm assuming the Unique Games Conjecture [4], and the best known approximation ratio is  $O(\sqrt{\log \text{opt}})$  due to Kratsch and Wahlström [24], improving the earlier bounds of  $O(\log n)$  [9] and  $O(\sqrt{\log n})$  [1]. It has been a long-standing open question whether BIPARTIZATION admits a PTAS on planar graphs or unit-disk graphs. On planar graphs, the currently best approximation for BIPARTIZATION is still the  $\frac{9}{4}$ -approximation algorithm given in the seminal work of Goemans and Williamson [15] more than two decades ago. This result immediately gives a 3-approximation algorithm for BIPARTIZATION on disk graphs (and in particular, unit-disk graphs) by the well-known fact that triangle-free disk graphs are planar [22].

► **Theorem 1** (folklore). *There exists a polynomial-time 3-approximation algorithm for BIPARTIZATION on the class of disk graphs.*

**Proof.** Let  $G$  be the input disk graph. We repeat the following step until  $G$  contains no triangles: find a triangle in  $G$ , add its three vertices to the solution, and remove them from  $G$ . Denote by  $S$  the set of vertices added to the solution and by  $G' = G - S$  the resulting triangle-free graph, which is planar [22]. Now apply the algorithm of Goemans and Williamson [15] on  $G'$  to obtain a  $\frac{9}{4}$ -approximation solution  $S'$ . Note that  $S \cup S'$  is a 3-approximation solution (for  $G$ ). Indeed, any solution of BIPARTIZATION must hit all triangles in  $G$  and thus contains at least  $|S|/3$  vertices in  $S$ . Also, it contains at least  $|S'|/4$  vertices in  $V(G')$ . So its size is at least  $|S|/3 + |S'|/4 \geq |S \cup S'|/3$ . ◀

For over two decades, this has remained the best approximation algorithm for BIPARTIZATION on disk graphs (in fact, even on unit-disk graphs). Thus, there is a natural question to be asked: can we achieve an approximation ratio better than 3 for the problem? Note that one cannot achieve this by improving the approximation ratio  $\frac{9}{4}$  for BIPARTIZATION on planar graphs. Indeed, even if we had a PTAS on planar graphs, the above argument still only gives us a 3-approximation algorithm on disk graphs. Therefore, the number 3 here is truly a bottleneck of the approximation ratio of the problem. In this paper, we break this bottleneck and answer the above question affirmatively by giving the first algorithm for BIPARTIZATION on disk graphs with an approximation ratio better than 3. Specifically, our main result is the following.

► **Theorem 2.** *There exists a polynomial-time  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on the class of disk graphs, for some constant  $\alpha > 0$ .*

We remark that Theorem 2 should be viewed as a proof of concept, rather than the quantitative improvement. Our algorithm in Theorem 2 is *robust* in the sense that it does not require the geometric realization of the input disk graph to be given. Furthermore, our algorithm directly generalizes to the broader class of *pseudo-disk* graphs, which are the intersection graphs of topological disks in which the boundaries of every two of them intersect at most twice. (Note that the 3-approximation algorithm in Theorem 1 also works for pseudo-disk graphs as triangle-free pseudo-disk graphs are planar [22].) Again, this generalized algorithm is robust.

► **Theorem 3.** *There exists a polynomial-time  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on the class of pseudo-disk graphs, for some constant  $\alpha > 0$ .*

Finally, we observe that our technique not only works for the BIPARTIZATION problem. In fact, it can be applied to a large category of vertex-deletion problems on (pseudo-)disk graphs, resulting in  $(3 - \alpha)$ -approximation algorithms. Although at this point it only yields interesting results for BIPARTIZATION (mainly because most well-studied problems in the category already have algorithms on disk graphs with approximation ratio better than 3), this demonstrates that our technique could possibly have further applications in the future. We shall briefly discuss this part in Section 4.

### Other related work

NP-complete optimization problems on disk graphs and other geometric intersection graphs have received considerable attention over years. Here we only summarize some recent work on this topic. The work of de Berg et al. [7] gave a framework for designing ETH-tight exact algorithms on (unit-)disk graphs or more generally (unit-)ball graphs, which works for a variety of classical optimization problems. Bonamy et al. [5] presented the first EPTAS and subexponential-time algorithm for MAXIMUM CLIQUE on disk graphs. Fomin et al. [11] designed almost ETH-tight parameterized algorithms for various cycle-packing and cycle-hitting problems on unit-disk graphs; in a follow-up paper [12], the same authors improved some of their algorithms to be ETH-tight. Recently, Lokshtanov et al. [26, 28] proposed frameworks for subexponential parameterized algorithms and EPTASes for various vertex-deletion problems on disk graphs (the framework for EPTASes does not work for BIPARTIZATION, while the one for subexponential parameterized algorithms works). A very recent work by the same authors [27] gave a 1.9999-approximation for VERTEX COVER on string graphs (i.e., intersection graphs of arbitrary connected geometric objects in the plane), which has the same flavor as this paper.

Besides the aforementioned algorithmic research on BIPARTIZATION, the problem was also studied in the context of kernelization complexity. The seminal work by Kratsch and Wahlström [23] showed that BIPARTIZATION admits a randomized polynomial kernel with respect to  $k$ . Later, for planar graphs, it was shown to admit a deterministic polynomial kernel by Jansen et al. [17]. Moreover, the kernelization complexity of BIPARTIZATION was studied also with respect to some structural parameterizations [16].

On a related note, we remark that structural properties of odd cycles in a graph has also received significant attention from various combinatorial points of view. While the survey of these results is beyond the scope of this paper, as an illustrative example, let us mention the study of Erdős–Pósa properties for odd cycles (see e.g., [10, 19, 33, 40]).

## 2 Preliminaries

Let  $G$  be a graph. We use  $V(G)$  and  $E(G)$  to denote the vertex set and edge set of  $G$ , respectively. For a subset  $V \subseteq V(G)$ , denote by  $G[V]$  the subgraph of  $G$  induced by  $V$ , and by  $G - V$  the subgraph of  $G$  induced by  $V(G) \setminus V$ . For a vertex  $v$ , we use  $N_G(v)$  to denote the set  $\{x \in V(G) \setminus \{v\} : (x, v) \in E(G)\}$ . For a vertex subset  $S$ , we use  $N_G(S)$  and  $N_G[S]$  to denote the sets  $\bigcup_{z \in S} N_G(z) \setminus S$  and  $S \cup N_G(S)$ , respectively. For a vertex  $v$  in  $G$ , we use  $d_G(v)$  to denote the degree of  $v$  (i.e.,  $|N_G(v)|$ ) in  $G$ . A vertex subset  $I \subseteq V(G)$  is a *distance- $d$  independent* set in  $G$ , if for any two distinct vertices  $x$  and  $y$  in  $I$ , the distance between  $x$  and  $y$  in  $G$  is strictly more than  $d$ . Here, the distance between two vertices is the number of edges in a shortest path between those vertices. Let  $\mathcal{S}$  be a collection of subsets of  $V(G)$ . A *packing* of  $\mathcal{S}$  is a sub-collection  $\mathcal{S}'$  such that  $S \cap S' = \emptyset$  for all  $S, S' \in \mathcal{S}'$  with  $S \neq S'$ . We say a packing  $\mathcal{S}' \subseteq \mathcal{S}$  is *maximal* if any  $\mathcal{S}'' \subseteq \mathcal{S}$  satisfying  $\mathcal{S}' \subsetneq \mathcal{S}''$  is not a packing of  $\mathcal{S}$ , and is *maximum* if any subset  $\mathcal{S}'' \subseteq \mathcal{S}$  satisfying  $|\mathcal{S}'| < |\mathcal{S}''|$  is not a packing of  $\mathcal{S}$ . Any maximum packing of  $\mathcal{S}$  has the same size.

An *odd cycle transversal* (or *OCT* for short) of  $G$  is a subset  $S \subseteq V(G)$  such that  $G - S$  is a bipartite graph. A *triangle* in  $G$  refers to a set  $T = \{u, v, w\}$  of three vertices of  $G$  such that  $(u, v), (v, w), (w, u) \in E(G)$ . We use  $\Delta(G)$  to denote the family of all triangles in  $G$ . For a set  $\mathcal{T}$  of triangles in  $G$ , we denote by  $V(\mathcal{T})$  the set of vertices of all triangles in  $\mathcal{T}$ , i.e.,  $V(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} T$ . The notation  $\text{tri}(\mathcal{T})$  denotes the size of a maximum packing of  $\mathcal{T}$ .

## 3 Our algorithm

In this section, we present our approximation algorithm for BIPARTIZATION on disk graphs. Our algorithm first takes a simple preprocessing step, which reduces the general problem to the problem on  $K_4$ -free disk graphs (i.e., disk graphs without cliques of size 4). Then in the main part of our algorithm, we solve BIPARTIZATION on  $K_4$ -free disk graphs. For a cleaner exposition, we shall present a randomized version of our algorithm, as it is more intuitive and yields a better approximation ratio. The derandomization can be found in the full version of the paper.

### 3.1 Preprocessing: reducing to the $K_4$ -free case

For a graph  $G$ , let  $K_4(G)$  be the set of all  $K_4$ 's in  $G$ . The following lemma allows us to reduce the problem to BIPARTIZATION on  $K_4$ -free disk graphs.

► **Lemma 4.** *Let  $G$  be a graph and  $\mathcal{C}$  be a packing of  $K_4(G)$ . Let  $S$  be a  $\rho$ -approximation solution for BIPARTIZATION on  $G - V(\mathcal{C})$ . Then,  $S \cup V(\mathcal{C})$  is a  $\max\{2, \rho\}$ -approximation solution for BIPARTIZATION on  $G$ .*



**Proof.** First, it is clear that  $S \cup V(\mathcal{C})$  is an OCT of  $G$ , since  $G - S \cup V(\mathcal{C}) = (G - V(\mathcal{C})) - S$  contains no odd cycles. Let  $S^*$  be an optimal OCT of  $G$ . For every  $K_4$  in  $\mathcal{C}$ ,  $S^*$  must contain at least 2 vertices in the  $K_4$  (for otherwise  $G - S^*$  contains a triangle). Since the  $K_4$ 's in  $\mathcal{C}$  are disjoint, we have  $|S^* \cap V(\mathcal{C})| \geq 2|\mathcal{C}| = |V(\mathcal{C})|/2$ . Furthermore,  $S^* \cap (V(G) \setminus V(\mathcal{C}))$  is an OCT of  $G - V(\mathcal{C})$ , which implies  $|S^* \cap (V(G) \setminus V(\mathcal{C}))| \geq |S|/\rho$ . Therefore,

$$|S^*| = |S^* \cap V(\mathcal{C})| + |S^* \cap (V(G) \setminus V(\mathcal{C}))| \geq \frac{|V(\mathcal{C})|}{2} + \frac{|S|}{\rho} \geq \frac{|S| + |V(\mathcal{C})|}{\max\{2, \rho\}}.$$

As  $|S \cup V(\mathcal{C})| = |S| + |V(\mathcal{C})|$ , we have  $|S \cup V(\mathcal{C})| \leq \max\{2, \rho\} \cdot |S^*|$ .  $\blacktriangleleft$

One reason for why this reduction helps us is the degeneracy of a  $K_4$ -free disk graph. Recall that a graph  $G$  is  $c$ -degenerate if we can sort its vertices as  $v_1, \dots, v_n$  such that each  $v_i$  is neighboring to at most  $c$  vertices in  $\{v_1, \dots, v_{i-1}\}$ . A  $c$ -degenerate graph of  $n$  vertices has at most  $cn$  edges and thus has average degree at most  $2c$ . We prove that every  $K_4$ -free disk graph is 11-degenerate. To prove this we use the following known result.

► **Lemma 5** ([30]). *Let  $D$  be a disk of radius  $r$ . Let  $\mathcal{S}$  be a set of pairwise non-overlapping disks of radius  $r$  such that every disk in  $\mathcal{S}$  intersects with  $D$ . Then,  $|\mathcal{S}| \leq 5$ .*

► **Lemma 6.** *Every  $K_4$ -free disk graph is 11-degenerate.*

**Proof.** To prove the lemma, it is enough to prove that for any  $K_4$ -free disk graph, there is a vertex of degree at most 11. Let  $G$  be a  $K_4$ -free disk graph with a realization  $\mathcal{D}$ , and let  $D_v \in \mathcal{D}$  be the disk representing the vertex  $v \in V(G)$ . Let  $u$  be a vertex in  $G$  such that disk  $D_u$  has the smallest radius among the disks in  $\mathcal{D}$ . We will prove that  $d_G(u) \leq 11$ . Let  $r$  be the radius of  $D_u$ . Notice that for each  $v \in N_G(u)$ , the radius of  $D_v$  is at least  $r$ . Now we construct a graph  $H$  with vertex set  $N_G[u]$  such that  $H$  is a unit disk graph,  $H$  is a subgraph of  $G$  (and hence  $K_4$ -free), and  $d_G(u) = d_H(u)$ . The construction of  $H$  is as follows. For each  $v \in N_G(u)$ , construct a disk  $D'_v$  of radius  $r$  which is fully contained in the disk  $D_v$  and intersects  $D_u$ . The graph  $H$  is the geometric intersection graph of  $\mathcal{D}' = \{D_u\} \cup \{D'_v : v \in N_G(u)\}$ . It is easy to see that  $H$  is a unit disk graph,  $H$  is a subgraph of  $G$  and  $d_G(u) = d_H(u)$ . For each  $v \in V(H) \setminus \{u\}$ , let  $L_v$  be the line segment between the centers of  $D_u$  and  $D'_v$ . Let  $\{v_1, \dots, v_t\} = N_G(u)$  such that the line segments  $L_{v_1}, L_{v_2}, \dots, L_{v_t}$  are in the clockwise order. We claim that  $t \leq 11$ . For the sake of contradiction assume that  $t \geq 12$ . Suppose there exists two distinct  $i, j \in \{1, 3, 5, 7, 9, 11\}$  such that  $D'_{v_i}$  intersects with  $D'_{v_j}$ . This implies that  $D'_{v_{i+1}}$  or  $D'_{v_{j+1}}$  intersects both  $D'_{v_i}$  and  $D'_{v_j}$ . Let  $w \in \{v_{i+1}, v_{j+1}\}$  be the vertex such that  $D'_w$  intersects both  $D'_{v_i}$  and  $D'_{v_j}$ . Then,  $H[\{u, w, v_i, v_j\}]$  is a complete graph on 4 vertices, which is a contradiction because  $H$  is  $K_4$ -free. Then, the disks  $D'_{v_1}, D'_{v_3}, D'_{v_5}, D'_{v_7}, D'_{v_9}, D'_{v_{11}}$  are pairwise non-overlapping, which is a contradiction to Lemma 5. Thus, we proved that  $t \leq 11$  and hence  $d_H(v) = d_G(v) \leq 11$ . That is, the degeneracy of  $G$  is at most 11.  $\blacktriangleleft$

### 3.2 The main algorithm

Our main algorithm for BIPARTIZATION on  $K_4$ -free disk graphs is presented in Algorithm 1. At the beginning, it takes an arbitrary maximal triangle packing  $\mathcal{T}$  of  $G$  (line 1) and defines  $\mathcal{O}$  as the triangles in  $G$  that have at least one vertex outside  $V(\mathcal{T})$ . In a high-level, our algorithm computes three different solutions  $S_1, S_2, S_3$  and returns the best one.

The first solution  $S_1$  is computed in exactly the same way as the 3-approximation algorithm described in the introduction. Specifically, we include in  $S_1$  all vertices in the triangle packing  $\mathcal{T}$ , and an OCT  $X_1$  of  $G - V(\mathcal{T})$  computed by a sub-routine PLANARBIP (line 3), which is an algorithm for BIPARTIZATION on planar graphs.

## 6:6 Bipartizing (Pseudo-)Disk Graphs

The second solution  $S_2$  is constructed in a more involved way. First, in each triangle  $T \in \mathcal{T}$ , we randomly sample one vertex  $v_T \in T$  (line 6); here the function  $\text{random}(T)$  returns each vertex in  $T$  with probability  $\frac{1}{3}$  and different calls of  $\text{random}$  are independent. Let  $R$  be the vertices in  $V(\mathcal{T})$  not sampled (line 7). Then we include in  $S_2$  all vertices in  $R$ , all vertices in an arbitrary maximal triangle packing  $\mathcal{T}'$  of  $G - R$  (line 8), and an OCT  $X_2$  of  $G - (R \cup V(\mathcal{T}'))$  computed by PLANARBIP.

If  $\mathcal{O}$  is nonempty, we need to construct our third solution  $S_3$ . We take a maximal packing  $\mathcal{T}''$  of the triangles in  $\mathcal{O}$  (line 12). Then we include in  $S_3$  all vertices in  $V(\mathcal{T}'') \cap V(\mathcal{T})$  and an OCT  $X_3$  of  $G - (V(\mathcal{T}'') \cap V(\mathcal{T}))$  recursively computed by our algorithm (line 13).

Finally, we return the best one among  $S_1, S_2, S_3$  (line 15); here  $\min\{S_1, S_2, S_3\}$  returns the set of smallest size among  $S_1, S_2, S_3$ . If  $S_3$  is not computed, we simply return  $\min\{S_1, S_2\}$ . It is obvious that each of  $S_1, S_2, S_3$  is an OCT of  $G$  and thus the algorithm always returns a correct solution. The quality of the solution obtained will be analyzed in the next section. Also, one can easily see that Algorithm 1 runs in polynomial time. Indeed, except the recursive call of OCT in line 13, all the other steps can be done in polynomial time. Line 13 will only be executed when  $\mathcal{O} \neq \emptyset$ . In this case,  $\mathcal{T}'' \neq \emptyset$  and  $V(\mathcal{T}'') \neq \emptyset$ . Thus, the graph  $G - (V(\mathcal{T}'') \cap V(\mathcal{T}))$  has at most  $n - 1$  vertices where  $n = V(G)$ , and the running time of Algorithm 1 satisfies the recurrence  $T(n) \leq T(n - 1) + n^{O(1)}$  which solves to  $T(n) = n^{O(1)}$ .

■ **Algorithm 1** BIPARTIZATION( $G$ ). ▷  $G$  is a  $K_4$ -free disk graph

---

```

1:  $\mathcal{T} \leftarrow$  a maximal packing of  $\Delta(G)$ 
2:  $\mathcal{O} \leftarrow \{T \in \Delta(G) : T \not\subseteq V(\mathcal{T})\}$ 
3:  $X_1 \leftarrow \text{PLANARBIP}(G - V(\mathcal{T}))$  ▷ construct the first solution  $S_1$ 
4:  $S_1 \leftarrow V(\mathcal{T}) \cup X_1$ 
5: for every  $T \in \mathcal{T}$  do ▷ construct the second solution  $S_2$ 
6:    $v_T \leftarrow \text{random}(T)$ 
7:    $R \leftarrow \bigcup_{T \in \mathcal{T}} (T \setminus \{v_T\})$ 
8:    $\mathcal{T}' \leftarrow$  a maximal packing of  $\Delta(G - R)$ 
9:    $X_2 \leftarrow \text{PLANARBIP}(G - (R \cup V(\mathcal{T}')))$ 
10:  $S_2 \leftarrow R \cup V(\mathcal{T}') \cup X_2$ 
11: if  $\mathcal{O} \neq \emptyset$  then ▷ construct the third solution  $S_3$ 
12:    $\mathcal{T}'' \leftarrow$  a maximal packing of  $\mathcal{O}$ 
13:    $X_3 \leftarrow \text{BIPARTIZATION}(G - (V(\mathcal{T}'') \cap V(\mathcal{T})))$ 
14:    $S_3 \leftarrow (V(\mathcal{T}'') \cap V(\mathcal{T})) \cup X_3$ 
15: return  $\min\{S_1, S_2, S_3\}$  ▷ if  $S_3$  is undefined, simply return  $\min\{S_1, S_2\}$ 

```

---

### 3.3 Analysis

In this section, we analyze the (expected) approximation ratio of Algorithm 1. We denote by  $\rho$  this ratio and aim to establish an upper bound for  $\rho$ . Consider a given disk graph  $G$  which is  $K_4$ -free. Let  $\text{opt}$  be the minimum size of an odd cycle transversal of  $G$ , and  $\mathcal{T}, \mathcal{O}$  be the two sets of triangles as in Algorithm 1. The output of Algorithm 1 is the best one among three OCT solutions  $S_1, S_2, S_3$ . Therefore, to analyze the approximation ratio of our algorithm, we have to consider the approximation ratios of  $S_1, S_2, S_3$ . It turns out that each solution  $S_i$  individually may be of size  $3\text{opt}$  or even larger in worst case. However, as we will see, the best one among them always admits an approximation ratio strictly smaller than 3.

In order to analyze the three solutions, we define two important parameters:  $a = |\mathcal{T}|/\text{opt}$  and  $b = \text{tri}(\mathcal{O})/\text{opt}$  (recall that  $\text{tri}(\mathcal{O})$  is the size of a *maximum* packing of  $\mathcal{O}$ ). Note that  $a, b \in [0, 1]$  because both  $|\mathcal{T}|$  and  $\text{tri}(\mathcal{O})$  are at most the size of a maximum triangle packing in  $G$ , which is smaller than or equal to  $\text{opt}$ . The analysis of  $S_1, S_2, S_3$  will be done in terms of  $a$  and  $b$ , that is, we shall represent the approximation ratios of  $S_1, S_2, S_3$  as functions of  $a$  and  $b$ . Roughly speaking, we shall show that  $S_1$  is good when  $a$  is small,  $S_2$  is good when  $b$  is small, and  $S_3$  is good when both  $a$  and  $b$  are large. For convenience, we use the notation  $\rho_0$  to denote the approximation ratio of the PLANARBIP sub-routine used in Algorithm 1.

### 3.3.1 The quality of $S_1$

The solution  $S_1$  is computed using exactly the 3-approximation algorithm described in the introduction. A more careful analysis shows that its approximation ratio is related to the parameter  $a$ : the smaller  $a$  is, the better  $S_1$  is.

► **Lemma 7.**  $|S_1| \leq (3a + \rho_0(1 - a)) \cdot \text{opt}$ .

**Proof.** Since  $a = |\mathcal{T}|/\text{opt}$ , and  $\mathcal{T}$  is a triangle packing, we get that  $|V(\mathcal{T})| = 3|\mathcal{T}| = 3a \cdot \text{opt}$ . Since  $\mathcal{T}$  is a triangle packing, any odd cycle transversal contains at least  $|\mathcal{T}|$  vertices from  $V(\mathcal{T})$ , the size of a minimum odd cycle transversal in  $G - V(\mathcal{T})$  is at most  $\text{opt} - |\mathcal{T}| = (1 - a)\text{opt}$ . Therefore,  $|S_1| = |V(\mathcal{T})| + |X_1| \leq (3a + \rho_0(1 - a)) \cdot \text{opt}$ . ◀

### 3.3.2 The quality of $S_2$

Figuring out the quality of  $S_2$  is the most involved part in our analysis. Basically, what we shall show is that whenever the parameter  $b$  is sufficiently small,  $S_2$  always gives us a better-than-3 approximation no matter what the value of  $a$  is. The analysis in this section shall explicitly use the fact that  $G$  is  $K_4$ -free. Let  $v_T, R$ , and  $\mathcal{T}'$  be as in Algorithm 1. We first need the following simple observation, which will allow us to bound the expected size of  $S_2$  using the expected size of  $\mathcal{T}'$ .

► **Observation 8.**  $\mathbb{E}[|R \cap S_{\text{opt}}|] \geq \frac{1}{3}|R|$  and  $|V(\mathcal{T}') \cap S_{\text{opt}}| \geq |\mathcal{T}'|$ .

**Proof.** Since  $S_{\text{opt}}$  is an OCT of  $G$ , it contains at least one vertex in every triangle  $T \in \mathcal{T}$ . Thus,  $\mathbb{E}[|(T \setminus \{v_T\}) \cap S_{\text{opt}}|] \geq \frac{2}{3}$ . By the linearity of expectation, we have

$$\mathbb{E}[|R \cap S_{\text{opt}}|] = \sum_{T \in \mathcal{T}} \mathbb{E}[|(T \setminus \{v_T\}) \cap S_{\text{opt}}|] \geq \frac{2}{3}|\mathcal{T}| = \frac{1}{3}|R|.$$

The fact  $|V(\mathcal{T}') \cap S_{\text{opt}}| \geq |\mathcal{T}'|$  follows from the fact that  $\mathcal{T}'$  is a triangle packing:  $S_{\text{opt}}$  contains at least one vertex in every triangle  $T \in \mathcal{T}'$ . ◀

The above observation implies that  $\mathbb{E}[|(R \cup V(\mathcal{T}') \cap S_{\text{opt}})|] \geq \frac{1}{3}|R| + \mathbb{E}[|\mathcal{T}'|]$ . Therefore, we have  $\mathbb{E}[|S_{\text{opt}} \setminus (R \cup V(\mathcal{T}'))|] \leq \text{opt} - \frac{1}{3}|R| - \mathbb{E}[|\mathcal{T}'|]$  and hence  $\mathbb{E}[|X_2|] \leq \rho_0 \cdot (\text{opt} - \frac{1}{3}|R| - \mathbb{E}[|\mathcal{T}'|])$ . It follows that

$$\begin{aligned} \mathbb{E}[|S_2|] &= |R| + \mathbb{E}[|V(\mathcal{T}')|] + \mathbb{E}[|X_2|] \\ &\leq |R| + 3 \cdot \mathbb{E}[|\mathcal{T}'|] + \rho_0 \cdot \left( \text{opt} - \frac{1}{3}|R| - \mathbb{E}[|\mathcal{T}'|] \right) \\ &= \left( 1 - \frac{\rho_0}{3} \right) \cdot |R| + \rho_0 \cdot \text{opt} + (3 - \rho_0) \cdot \mathbb{E}[|\mathcal{T}'|] \end{aligned} \tag{1}$$

## 6:8 Bipartizing (Pseudo-)Disk Graphs

We say a vertex  $v \in V(\mathcal{T})$  is *dead* if  $v \notin R$  and  $v$  is not contained in any triangle in  $G[V(\mathcal{T}) \setminus R]$ . Let  $D$  denote the set of all dead vertices, which is a random subset of  $V(\mathcal{T})$  as it depends on the random vertices  $v_T$ . For each  $v \in V(\mathcal{T})$ , let  $\deg(v)$  denote the degree of  $v$  in  $G[V(\mathcal{T})]$ . Recall that  $a = |\mathcal{T}|/\text{opt}$  and  $b = \text{tri}(\mathcal{O})/\text{opt}$ . It is easy to see the following relation between  $|\mathcal{T}'|$  and  $|D|$ .

► **Observation 9.**  $|\mathcal{T}'| \leq (a + b) \cdot \text{opt} - \frac{|R|}{3} - \frac{|D|}{3}$ .

**Proof.** Since  $\mathcal{T}'$  is a triangle packing, we have  $|\mathcal{T}' \cap \mathcal{O}| \leq \text{tri}(\mathcal{O}) = b \cdot \text{opt}$ . On the other hand, all elements in  $\mathcal{T}' \setminus \mathcal{O}$  are triangles in  $G[V(\mathcal{T}) \setminus R]$ . However, dead vertices cannot be the vertex of any triangle in  $G[V(\mathcal{T}) \setminus R]$ . Thus, the vertices of the triangles in  $\mathcal{T}' \setminus \mathcal{O}$  must lie in  $V(\mathcal{T}) \setminus (R \cup D)$ . We have  $|V(\mathcal{T}) \setminus (R \cup D)| = 3 \cdot |\mathcal{T}| - |R| - |D| = 3a \cdot \text{opt} - |R| - |D|$ , which implies  $|\mathcal{T}' \setminus \mathcal{O}| \leq a \cdot \text{opt} - \frac{|R|}{3} - \frac{|D|}{3}$ . Because  $|\mathcal{T}'| = |\mathcal{T}' \cap \mathcal{O}| + |\mathcal{T}' \setminus \mathcal{O}|$ , we have the inequality in the observation. ◀

Combining the above observation with Equation 1, we have

$$\begin{aligned} \mathbb{E}[|S_2|] &\leq \left(1 - \frac{\rho_0}{3}\right) \cdot |R| + \rho_0 \cdot \text{opt} + (3 - \rho_0) \cdot \mathbb{E}[|\mathcal{T}'|] \\ &= \left(1 - \frac{\rho_0}{3}\right) \cdot |R| + \rho_0 \cdot \text{opt} + (3 - \rho_0) \cdot \left((a + b) \cdot \text{opt} - \frac{|R|}{3} - \frac{\mathbb{E}[|D|]}{3}\right) \\ &= (\rho_0 + (3 - \rho_0)(a + b)) \cdot \text{opt} - (3 - \rho_0) \cdot \frac{\mathbb{E}[|D|]}{3}. \end{aligned} \quad (2)$$

To show that  $S_2$  has an approximation ratio below 3 when  $b$  is small, the crucial observation is that we have a large number of dead vertices in expectation.

► **Observation 10.**  $\Pr[v \text{ is dead}] \geq \left(\frac{1}{3}\right)^{\frac{3}{8}\deg(v) + \frac{1}{4}}$  for all  $v \in V(\mathcal{T})$ . Thus, we have  $\mathbb{E}[|D|] \geq \left(\frac{1}{3}\right)^{\frac{3}{8}d + \frac{1}{4}}(3a \cdot \text{opt})$ , where  $d$  is the average degree of  $G[V(\mathcal{T})]$ .

**Proof.** Let  $v \in V(\mathcal{T})$  and  $T_0 \in \mathcal{T}$  be the triangle containing  $v$ . Denote by  $N(v)$  the set of neighbors of  $v$  in  $V(\mathcal{T})$  (excluding  $v$  itself). We then have  $|N(v)| = \deg(v)$  and  $|N(v) \setminus T_0| = \deg(v) - 2$ . Observe that the graph  $G[N(v)]$  is triangle-free. Indeed, if  $G[N(v)]$  contains a triangle  $T$ , then  $T \cup \{v\}$  forms a clique in  $G$  of size 4, which contradicts with the fact that  $G$  is  $K_4$ -free. As  $G[N(v)]$  is triangle-free, it is planar. In particular,  $G[N(v) \setminus T_0]$  is planar. It was known that every  $n$ -vertex planar graph has a vertex cover of size at most  $\frac{3}{4}n$  (indeed, a planar graph is 4-colorable, so it has an independent set of size at least  $\frac{n}{4}$  and thus a vertex cover of size at most  $\frac{3}{4}n$ ). Thus,  $G[N(v) \setminus T_0]$  has a vertex cover  $C \subseteq N(v) \setminus T_0$  with  $|C| \leq \frac{3}{4}|N(v) \setminus T_0| = \frac{3}{4}(\deg(v) - 2)$ .

Next, we notice that if  $v \notin R$  and  $C \subseteq R$ , then  $v$  is a dead vertex. Indeed, if  $v$  is contained in a triangle  $\{v, u, w\}$  in  $G[V(\mathcal{T}) \setminus R]$ , then at least one of  $u$  and  $w$  must be in  $C$ , since  $C$  is a vertex cover of  $N(v) \setminus T_0$  (note that  $u, w \notin T_0$  for otherwise  $v \in R$ ). Let  $\mathcal{T}_1 \subseteq \mathcal{T}$  (resp.,  $\mathcal{T}_2 \subseteq \mathcal{T}$ ) consist of the triangles that contain one vertex (resp., two vertices) in  $C$ . Note that no triangle in  $\mathcal{T}$  can contain three vertices in  $C$  because  $G[C]$  is triangle-free. Therefore,  $C \subseteq R$  if and only if  $v_T \notin C$  for all  $T \in \mathcal{T}_1 \cup \mathcal{T}_2$ . The events  $v_T \notin C$  for all  $T \in \mathcal{T}_1 \cup \mathcal{T}_2$  are independent, and happen with probability  $\frac{2}{3}$  if  $T \in \mathcal{T}_1$  and with probability  $\frac{1}{3}$  if  $T \in \mathcal{T}_2$ . It follows that  $\Pr[C \subseteq R] = \left(\frac{2}{3}\right)^{|\mathcal{T}_1|} \cdot \left(\frac{1}{3}\right)^{|\mathcal{T}_2|}$ . We have the inequality  $|\mathcal{T}_1| + 2|\mathcal{T}_2| = |C| \leq \frac{3}{4}(\deg(v) - 2)$ . Subject to  $|\mathcal{T}_1| \geq 0$ ,  $|\mathcal{T}_2| \geq 0$ , and  $|\mathcal{T}_1| + 2|\mathcal{T}_2| \leq \frac{3}{4}(\deg(v) - 2)$ , the quantity  $\left(\frac{2}{3}\right)^{|\mathcal{T}_1|} \cdot \left(\frac{1}{3}\right)^{|\mathcal{T}_2|}$  is minimized when  $|\mathcal{T}_1| = 0$  and  $|\mathcal{T}_2| = \frac{3}{8}(\deg(v) - 2)$ , and is equal to  $\left(\frac{1}{3}\right)^{\frac{3}{8}(\deg(v) - 2)}$ . Thus, we have  $\Pr[C \subseteq R] \geq \left(\frac{1}{3}\right)^{\frac{3}{8}(\deg(v) - 2)}$ . Furthermore, the events  $v \notin R$  and  $C \subseteq R$  are independent because whether  $v \notin R$  happens only depends on the choice of  $v_{T_0} \in T_0$ . We have  $\Pr[v \notin R] = \frac{1}{3}$  and  $\Pr[C \subseteq R] \geq \left(\frac{1}{3}\right)^{\frac{3}{8}(\deg(v) - 2)}$ . Since  $v$  is a dead vertex if both  $v \notin R$  and  $C \subseteq R$  happen, we finally have  $\Pr[v \text{ is dead}] \geq \left(\frac{1}{3}\right)^{\frac{3}{8}\deg(v) + \frac{1}{4}}$ . By the linearity of expectation and the fact  $|V(\mathcal{T})| = 3|\mathcal{T}| = 3a \cdot \text{opt}$ , we then have

$$\mathbb{E}[|D|] = \sum_{v \in V(\mathcal{T})} \Pr[v \text{ is dead}] \geq \sum_{v \in V(\mathcal{T})} \left(\frac{1}{3}\right)^{\frac{3}{8}\deg(v) + \frac{1}{4}} \geq \left(\frac{1}{3}\right)^{\frac{3}{8}d + \frac{1}{4}} (3a \cdot \text{opt}).$$

where  $d = \sum_{v \in V(\mathcal{T})} \deg(v) / |V(\mathcal{T})|$ . ◀

At the end, we can establish the bound for  $\mathbb{E}[|S_2|]$ .

► **Lemma 11.**  $\mathbb{E}[|S_2|] \leq (\rho_0 + (1 - (\frac{1}{3})^{\frac{3}{8}d + \frac{1}{4}})(3 - \rho_0)a + (3 - \rho_0)b) \cdot \text{opt}$ , where  $d$  denotes the average degree of  $G[V(\mathcal{T})]$ .

**Proof.** Combining Observation 10 with Equation 2 completes the proof. ◀

According to Lemma 6, we have  $d \leq 22$ , and thus the above lemma gives us a good bound for  $\mathbb{E}[|S_2|]$ : as long as  $b$  is sufficiently small, no matter what  $a$  is, we can have that  $\mathbb{E}[|S_2|] \leq (3 - \alpha) \cdot \text{opt}$  for some constant  $\alpha > 0$ .

### 3.3.3 The quality of $S_3$

Finally, we analyze the quality of  $S_3$ . Given  $S_1$  is good when  $a$  is small and  $S_2$  is good when  $b$  is small, we clearly want  $S_3$  to be good when both  $a$  and  $b$  are large.

► **Lemma 12.** *If  $\rho \geq 2$ , then  $\mathbb{E}[|S_3|] \leq (\frac{2b}{3} + \rho(2 - a - \frac{b}{3})) \cdot \text{opt}$ .*

**Proof.** Let  $r = |\mathcal{T}''| / \text{opt}$ . Since  $\mathcal{T}''$  is a *maximal* packing in  $\mathcal{O}$ ,  $V(\mathcal{T}'')$  is a hitting set of  $\mathcal{O}$ , which implies  $3|\mathcal{T}''| = |V(\mathcal{T}'')| \geq \text{tri}(\mathcal{O})$  and thus  $r \geq \frac{\text{tri}(\mathcal{O})}{3 \cdot \text{opt}} = \frac{b}{3}$ .

▷ **Claim 13.**  $\mathbb{E}[|X_3|] \leq \rho(2 - a - r)\text{opt}$ .

**Proof.** Let  $S^*$  be an optimal OCT of  $G$ . So  $|S^*| = \text{opt}$ . We call a triangle  $T \in \mathcal{T}''$  *bad* if  $S^*$  contains a vertex from  $V(T) \setminus V(\mathcal{T})$ . Since  $\mathcal{T}$  is a triangle packing, we also know that  $|S^* \cap V(\mathcal{T})| \geq |\mathcal{T}| = a \cdot \text{opt}$ . Thus, the number of bad triangles in  $\mathcal{T}''$  is at most  $(1 - a)\text{opt}$ , because  $S^*$  contains at most  $(1 - a)\text{opt}$  vertices outside of  $V(\mathcal{T})$ , and any such vertex can be part of at most one triangle in  $\mathcal{T}''$ . That is, the number of *good* triangles in  $\mathcal{T}''$  is at least  $(r - (1 - a))\text{opt}$ . For each good triangle  $T \in \mathcal{T}''$ ,  $S^*$  contains a vertex from  $V(T) \cap V(\mathcal{T})$ . This implies that  $|S^* \cap (V(\mathcal{T}'') \cap V(\mathcal{T}))| \geq (r - (1 - a))\text{opt}$ , and hence  $|S^* \setminus (V(\mathcal{T}'') \cap V(\mathcal{T}))| \leq \text{opt} - (r - (1 - a))\text{opt} = (2 - a - r)\text{opt}$ . Also, notice that  $S^* \setminus (V(\mathcal{T}'') \cap V(\mathcal{T}))$  is an OCT of  $G - (V(\mathcal{T}'') \cap V(\mathcal{T}))$ . Thus, the size of an optimal OCT of  $G - (V(\mathcal{T}'') \cap V(\mathcal{T}))$  is at most  $(2 - a - r)\text{opt}$ , which implies  $\mathbb{E}[|X_3|] \leq \rho(2 - a - r)\text{opt}$ . ◀

Since each triangle  $T \in \mathcal{T}''$  is also in  $\mathcal{O}$ , we have  $|V(T) \cap V(\mathcal{T})| \leq 2$ . This implies that  $|V(\mathcal{T}'') \cap V(\mathcal{T})| \leq 2r \cdot \text{opt}$ . Now we are ready to deduce

$$\begin{aligned} \mathbb{E}[|S_3|] &= \mathbb{E}[|V(\mathcal{T}'') \cap V(\mathcal{T})| + |X_3|] \\ &= |V(\mathcal{T}'') \cap V(\mathcal{T})| + \mathbb{E}[|X_3|] \\ &\leq (2r \cdot \text{opt}) + \rho(2 - a - r)\text{opt} \quad (\text{By Claim 13 and } |V(\mathcal{T}'') \cap V(\mathcal{T})| \leq 2r \cdot \text{opt}) \\ &= (2 - \rho)r \cdot \text{opt} + \rho(2 - a)\text{opt} \\ &\leq (2 - \rho)\frac{b}{3}\text{opt} + \rho(2 - a)\text{opt} \quad (\text{Because } r \geq \frac{b}{3}, \text{ and } 2 - \rho < 0) \\ &\leq \left(\frac{2b}{3} + \rho\left(2 - a - \frac{b}{3}\right)\right) \cdot \text{opt}. \end{aligned}$$

This completes the proof of the lemma. ◀

### 3.3.4 Putting everything together

Given the analyses for  $S_1$ ,  $S_2$ , and  $S_3$  in the previous sections, we are ready to bound the (expected) approximation ratio  $\rho$  of the entire algorithm. Let  $\rho_1 = 3a + \rho_0(1 - a)$ ,  $\rho_2 = \rho_0 + (1 - (\frac{1}{3})^{\frac{3}{8}d + \frac{1}{4}})(3 - \rho_0)a + (3 - \rho_0)b$ ,  $\rho_3 = \frac{2b}{3} + \rho(2 - a - \frac{b}{3})$  be the approximation ratios of  $S_1, S_2, S_3$  given in Lemmas 7, 11, 12, respectively<sup>1</sup>. As the output is the best one among  $S_1, S_2, S_3$ , we have

$$\rho \leq \frac{\mathbb{E}[\min\{|S_1|, |S_2|, |S_3|\}]}{\text{opt}} \leq \frac{\min\{\mathbb{E}[|S_1|], \mathbb{E}[|S_2|], \mathbb{E}[|S_3|]\}}{\text{opt}} \leq \min\{\rho_1, \rho_2, \rho_3\}.$$

Note that  $\rho_1, \rho_2, \rho_3$  can be viewed as linear functions of  $a$  and  $b$ , when the other numbers  $d, \rho_0, \rho$  are all fixed. So we first figure out the values of  $a$  and  $b$  that maximizes  $\min\{\rho_1, \rho_2, \rho_3\}$ . With calculation, we have

$$\min\{\rho_1, \rho_2, \rho_3\} \leq \frac{(3 - \rho_0)(2\rho - \rho_0)}{(3 - \rho_0 + \rho) + (\rho - 2) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})}} + \rho_0, \quad (3)$$

and the upper bound is achieved when

$$a = \frac{(2\rho - \rho_0)}{(3 - \rho_0 + \rho) + (\rho - 2) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})}} \quad \text{and} \quad b = \frac{(2\rho - \rho_0) \cdot 3^{\frac{3}{8}d + \frac{1}{4}}}{(3 - \rho_0 + \rho) + (\rho - 2) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})}},$$

in which case  $\rho_1 = \rho_2 = \rho_3$ . Now combine Equation 3 with the inequality  $\rho \leq \min\{\rho_1, \rho_2, \rho_3\}$  and re-arrange the terms in the inequality, we deduce

$$(3^{-(\frac{3}{8}d + \frac{5}{4})} + 1) \cdot \rho^2 - ((2 + \rho_0) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})} + 3) \cdot \rho + (2\rho_0 \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})}) \leq 0.$$

The left-hand side of the above inequality is a quadratic function of  $\rho$  in which the coefficient of the quadratic term is positive. Therefore, in order to make the quadratic function non-positive,  $\rho$  must be smaller than its larger root, i.e.,

$$\rho \leq \frac{((2 + \rho_0) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})} + 3) + \sqrt{((2 + \rho_0) \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})} + 3)^2 - (3^{-(\frac{3}{8}d + \frac{5}{4})} + 1)(8\rho_0 \cdot 3^{-(\frac{3}{8}d + \frac{5}{4})})}}{2 \cdot (3^{-(\frac{3}{8}d + \frac{5}{4})} + 1)}.$$

By Lemma 6,  $G[V(\mathcal{T})]$  is 11-degenerate and thus  $d \leq 22$ . Furthermore, using the  $\frac{9}{4}$ -approximation algorithm [15] for planar bipartization, we can set  $\rho_0 = \frac{9}{4}$ . Plugging in these values to the above inequality, we have  $\rho \leq 2.99993033741$ .

Our entire algorithm first applies Lemma 4 with a maximal packing  $\mathcal{C}$  of  $K_4(G)$  to reduce the problem to  $G - V(\mathcal{C})$ , which is a  $K_4$ -free disk graph, and then applies Algorithm 1 on  $G - V(\mathcal{C})$ . By Lemma 4 and the above analysis, this algorithm solves BIPARTIZATION on disk graphs with an *expected* approximation ratio at most 2.99993033741. By repeating the algorithm polynomial number of times, we can also obtain a randomized algorithm that achieves the same approximation ratio with high probability.

► **Theorem 14.** *There exists a polynomial-time randomized algorithm for BIPARTIZATION on the class of disk graphs that gives a  $(3 - \alpha)$ -approximation solution with high probability, for some  $\alpha > 10^{-5}$ .*

<sup>1</sup> In Lemma 12, the bound for  $\mathbb{E}[|S_3|]$  has a condition  $\rho \geq 2$ . But we can assume this is always the case, for otherwise our algorithm is already a 2-approximation algorithm.

With some efforts, we can derandomize our algorithm to obtain a deterministic  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on disk graphs. In fact, in Algorithm 1, only the construction of the set  $R$  is randomized. We show in the full version how to construct  $R$  deterministically while still guaranteeing the nice properties of  $R$  (the key point is to guarantee that there are many dead vertices). The approximation ratio of our deterministic algorithm is slightly worse than the randomized one (while it is still smaller than 3).

► **Theorem 2.** *There exists a polynomial-time  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on the class of disk graphs, for some constant  $\alpha > 0$ .*

## 4 Generalizations

In this section, we discuss some generalizations of our result. First, we observe that our algorithm directly generalizes to *pseudo-disk graphs*. A set of geometric objects in the plane are called *pseudo-disks* if each of them is homeomorphic to a disk and the boundaries of any two objects intersect at most twice. A graph is a *pseudo-disk graph* if it can be represented as the intersection graph of a set of pseudo-disks.

In our BIPARTIZATION algorithm, we only exploit two properties of disk graphs: (i) triangle-free disk graphs are planar, and (ii)  $K_4$ -free disk graphs are  $c$ -degenerate for some constant  $c$ . In fact, pseudo-disk graphs also satisfy these two properties.

► **Fact 15** ([22]). *Triangle-free pseudo-disk graphs are planar.*

► **Fact 16.**  *$K_4$ -free pseudo-disk graphs are  $c$ -degenerate for some constant  $c$ .*

**Proof.** We show that any  $K_r$ -free pseudo-disk graph of  $n$  vertices only has  $O(rn)$  edges, which implies the fact. Let  $G$  be a  $K_r$ -free pseudo-disk graph realized by a set  $\mathcal{S}$  of pseudo-disks. We say an edge  $(S, S')$  of  $G$  is an *inclusion edge* if  $S \subseteq S'$  or  $S' \subseteq S$ . We first observe that  $G$  has  $O(rn)$  inclusion edges. Indeed, a pseudo-disk  $S \in \mathcal{S}$  cannot be contained in  $r - 1$  (or more) other pseudo-disks in  $\mathcal{S}$ , for otherwise there is a copy of  $K_r$  in  $G$ . Thus, if we charge every inclusion edge  $(S, S')$  to the smaller one of  $S$  and  $S'$ , every pseudo-disk is charged at most  $r - 2$  times. This implies that  $G$  has  $O(rn)$  inclusion edges. Now we bound the number of other edges in  $G$ . Note that if  $(S, S')$  is a non-inclusion edge in  $G$ , then the boundaries of  $S$  and  $S'$  intersect. So it suffices to bound the total number of intersection points of the boundaries of the pseudo-disks in  $\mathcal{S}$ . The depth of an intersection point  $x$  is the number of pseudo-disks in  $\mathcal{S}$  containing  $x$ . It is well-known that in a set of  $n$  pseudo-disks, the number of boundary intersection points of depth at most  $d$  is bounded by  $O(dn)$  [21]. Since  $G$  is  $K_r$ -free, every intersection point is of depth at most  $r$ . Thus, the total number of boundary intersection points is  $O(rn)$ , implying that  $G$  has  $O(rn)$  edges. ◀

Therefore, our algorithm directly generalizes to pseudo-disk graphs.

► **Theorem 3.** *There exists a polynomial-time  $(3 - \alpha)$ -approximation algorithm for BIPARTIZATION on the class of pseudo-disk graphs, for some constant  $\alpha > 0$ .*

Next, we observe that our techniques apply to not only the specific problem of BIPARTIZATION. In fact, it works for a wide class of vertex-deletion problems on (pseudo-)disk graphs. Recall that in a vertex-deletion problem, the goal is to delete a minimum set  $S$  of vertices from a graph  $G$  such that  $G - S$  satisfies some desired property  $\mathbf{P}$ . In BIPARTIZATION, the property  $\mathbf{P}$  is “being bipartite”. Our technique applies to any vertex-deletion problem that is (i) hereditary, i.e., if a graph satisfies  $\mathbf{P}$  then all its induced subgraphs also satisfy  $\mathbf{P}$ , and (ii) triangle-conflicting, i.e., a graph satisfies  $\mathbf{P}$  only if it is triangle-free.

► **Theorem 17.** *If a hereditary and triangle-conflicting vertex-deletion problem admits a  $(3 - \delta)$ -approximation algorithm on the class of planar graphs for some  $\delta > 0$ , then it admits a  $(3 - \alpha)$ -approximation algorithm on the class of (pseudo-)disk graphs for some  $\alpha > 0$ .*

**Proof.** We just replace the PLANARBIP sub-routine in Algorithm 1 with the  $(3 - \delta)$ -approximation algorithm for the problem on planar graphs. As one can easily verify, our analysis only depends on the fact that the vertex-deletion problem is hereditary and triangle-conflicting. Thus, the same analysis shows that this is a  $(3 - \alpha)$ -approximation algorithm for the problem on (pseudo-)disk graphs. ◀

Well-studied instances of hereditary and triangle-conflicting vertex-deletion problems (other than BIPARTIZATION) include VERTEX COVER, FEEDBACK VERTEX SET, TRIANGLE HITTING, etc. Most of these problems already have  $(3 - \delta)$ -approximation algorithms on disk graphs, and some of them do not have known  $(3 - \delta)$ -approximation algorithms on planar graphs. Thus, at this point, we can only obtain interesting results for BIPARTIZATION. However, we believe that this is not the full power of Theorem 17. To provide some evidences, let us consider a vertex-deletion problem, PLANARIZATION&BIPARTIZATION, in which the desired property  $\mathbf{P}$  is “being planar and bipartite”. This problem is clearly hereditary and triangle-conflicting. On planar graphs, it is equivalent to BIPARTIZATION and thus admits a  $(3 - \delta)$ -approximation algorithm. Therefore, Theorem 17 gives a  $(3 - \alpha)$ -approximation algorithm for PLANARIZATION&BIPARTIZATION on (pseudo-)disk graphs. Although this problem itself is somehow artificial and not well-studied, it reveals that Theorem 17 could possibly have further applications in the future.

---

## 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 Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 573–581. ACM, 2005. doi:10.1145/1060590.1060675.
- 2 Sayan Bandyapadhyay, William Lochet, Daniel Lokshtanov, Saket Saurabh, and Jie Xue. Subexponential parameterized algorithms for cut and cycle hitting problems on h-minor-free graphs. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2063–2084. SIAM, 2022.
- 3 Sayan Bandyapadhyay, William Lochet, Daniel Lokshtanov, Saket Saurabh, and Jie Xue. True contraction decomposition and almost ETH-tight bipartization for unit-disk graphs. In *38th International Symposium on Computational Geometry (SoCG 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 4 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.
- 5 Marthe Bonamy, Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Panos Giannopoulos, Eun Jung Kim, Pawel Rzazewski, Florian Sikora, and Stéphan Thomassé. EPTAS and subexponential algorithm for maximum clique on disk and unit ball graphs. *J. ACM*, 68(2):9:1–9:38, 2021. doi:10.1145/3433160.
- 6 Hyeon-Ah Choi, Kazuo Nakajima, and Chong S. Rim. Graph bipartization and via minimization. *SIAM J. Discret. Math.*, 2(1):38–47, 1989. doi:10.1137/0402004.
- 7 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.





- 8 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.
- 9 Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000. doi:10.1145/347476.347478.
- 10 Samuel Fiorini, Nadia Hardy, Bruce A. Reed, and Adrian Vetta. Approximate min-max relations for odd cycles in planar graphs. *Math. Program.*, 110(1):71–91, 2007. doi:10.1007/s10107-006-0063-7.
- 11 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discret. Comput. Geom.*, 62(4):879–911, 2019. doi:10.1007/s00454-018-00054-x.
- 12 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. ETH-tight algorithms for long path and cycle on unit disk graphs. *J. Comput. Geom.*, 12(2):126–148, 2021. doi:10.20382/jocg.v12i2a6.
- 13 Matt Gibson and Imran A. Pirwani. Algorithms for dominating set in disk graphs: Breaking the  $\log n$  barrier – (extended abstract). In Mark de Berg and Ulrich Meyer, editors, *Algorithms – ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, volume 6346 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2010. doi:10.1007/978-3-642-15775-2\_21.
- 14 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, 1995. doi:10.1145/227683.227684.
- 15 Michel X. Goemans and David P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Comb.*, 18(1):37–59, 1998. doi:10.1007/PL00009810.
- 16 Bart MP Jansen and Stefan Kratsch. On polynomial kernels for structural parameterizations of odd cycle transversal. In *International Symposium on Parameterized and Exact Computation*, pages 132–144. Springer, 2011.
- 17 Bart MP Jansen, Marcin L Pilipczuk, and Erik Jan Van Leeuwen. A deterministic polynomial kernel for odd cycle transversal and vertex multiway cut in planar graphs. *SIAM Journal on Discrete Mathematics*, 35(4):2387–2429, 2021.
- 18 Andrew B. Kahng, Shailesh Vaya, and Alexander Zelikovsky. New graph bipartizations for double-exposure, bright field alternating phase-shift mask layout. In Satoshi Goto, editor, *Proceedings of ASP-DAC 2001, Asia and South Pacific Design Automation Conference 2001, January 30-February 2, 2001, Yokohama, Japan*, pages 133–138. ACM, 2001. doi:10.1145/370155.370304.
- 19 Ken-Ichi Kawarabayashi and Atsuhiko Nakamoto. The Erdős–Pósa property for vertex-and edge-disjoint odd cycles in graphs on orientable surfaces. *Discrete Mathematics*, 307(6):764–768, 2007.
- 20 Ken-ichi Kawarabayashi and Bruce A. Reed. Odd cycle packing. 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 695–704. ACM, 2010. doi:10.1145/1806689.1806785.
- 21 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1):59–71, 1986.
- 22 Jan Kratochvíl. Intersection graphs of noncrossing arc-connected sets in the plane. In *International Symposium on Graph Drawing*, pages 257–270. Springer, 1996.
- 23 Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms (TALG)*, 10(4):1–15, 2014.
- 24 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi:10.1145/3390887.

- 25 Daniel Lokshantov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.
- 26 Daniel Lokshantov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031. SIAM, 2022.
- 27 Daniel Lokshantov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A 1.9999-approximation algorithm for vertex cover on string graphs. In *40th International Symposium on Computational Geometry (SoCG 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 28 Daniel Lokshantov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. A framework for approximation schemes on disk graphs. In *34rd Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2023.
- 29 Daniel Lokshantov, Saket Saurabh, and Magnus Wahlström. Subexponential parameterized odd cycle transversal on planar graphs. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.
- 30 Madhav V. Marathe, Heinz Breu, Harry B. Hunt, S. S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.
- 31 Dániel Marx, Pranabendu Misra, Daniel Neuen, and Prafullkumar Tale. A framework for parameterized subexponential algorithms for generalized cycle hitting problems on planar graphs. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2085–2127. SIAM, 2022.
- 32 Mihai Pop, Daniel S Kosack, and Steven L Salzberg. Hierarchical scaffolding with bambus. *Genome research*, 14(1):149–159, 2004.
- 33 Dieter Rautenbach and Bruce Reed. The Erdos-Pósa property for odd cycles in highly connected graphs. *Combinatorica*, 21(2):267–278, 2001.
- 34 Dieter Rautenbach and Bruce A. Reed. The Erdos-Pósa property for odd cycles in highly connected graphs. *Comb.*, 21(2):267–278, 2001. doi:10.1007/s004930100024.
- 35 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.
- 36 Bruce A. Reed. Mangoes and blueberries. *Comb.*, 19(2):267–296, 1999. doi:10.1007/s004930050056.
- 37 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.
- 38 Romeo Rizzi, Vineet Bafna, Sorin Istrail, and Giuseppe Lancia. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In Roderic Guigó and Dan Gusfield, editors, *Algorithms in Bioinformatics, Second International Workshop, WABI 2002, Rome, Italy, September 17-21, 2002, Proceedings*, volume 2452 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2002. doi:10.1007/3-540-45784-4\_3.
- 39 Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976. doi:10.1145/321958.321975.
- 40 Carsten Thomassen. The Erdős-Pósa property for odd cycles in graphs of large connectivity. *Combinatorica*, 21(2):321–333, 2001.
- 41 Erik Jan van Leeuwen. Better approximation schemes for disk graphs. In Lars Arge and Rusins Freivalds, editors, *Algorithm Theory – SWAT 2006, 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006, Proceedings*, volume 4059 of *Lecture Notes in Computer Science*, pages 316–327. Springer, 2006. doi:10.1007/11785293\_30.
- 42 Sebastian Wernicke. *On the algorithmic tractability of single nucleotide polymorphism (SNP) analysis and related problems*. diplom. de, 2014.

# A Logarithmic Approximation of Linearly-Ordered Colourings

Johan Håstad   

KTH Royal Institute of Technology, Stockholm, Sweden

Björn Martinsson  

KTH Royal Institute of Technology, Stockholm, Sweden

Tamio-Vesa Nakajima   

Department of Computer Science, University of Oxford, UK

Stanislav Živný   

Department of Computer Science, University of Oxford, UK

---

## Abstract

A linearly ordered (LO)  $k$ -colouring of a hypergraph assigns to each vertex a colour from the set  $\{0, 1, \dots, k - 1\}$  in such a way that each hyperedge has a unique maximum element. Barto, Batistelli, and Berg conjectured that it is NP-hard to find an LO  $k$ -colouring of an LO 2-colourable 3-uniform hypergraph for any constant  $k \geq 2$  [STACS'21] but even the case  $k = 3$  is still open. Nakajima and Živný gave polynomial-time algorithms for finding, given an LO 2-colourable 3-uniform hypergraph, an LO colouring with  $O^*(\sqrt{n})$  colours [ICALP'22] and an LO colouring with  $O^*(\sqrt[3]{n})$  colours [ACM ToCT'23]. Very recently, Louis, Newman, and Ray gave an SDP-based algorithm with  $O^*(\sqrt[5]{n})$  colours. We present two simple polynomial-time algorithms that find an LO colouring with  $O(\log_2(n))$  colours, which is an exponential improvement.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Constraint and logic programming

**Keywords and phrases** Linear ordered colouring, Hypergraph, Approximation, Promise Constraint Satisfaction Problems

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.7

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2404.19556>

**Funding** This work was supported by UKRI EP/X024431/1, by a Clarendon Fund Scholarship and by the Knut and Alice Wallenberg Foundation. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. All data is provided in full in the results section of this paper.

**Acknowledgements** This paper is a merger of independent work by Håstad and Martinsson, and by Nakajima and Živný respectively. We are grateful to Venkat Guruswami for noting and informing the authors of the fact that we independently had found the same algorithm.

## 1 Introduction

Given a graph  $G$ , the *graph  $k$ -colouring* problem asks to find a colouring of the vertices of  $G$  by colours from the set  $\{0, 1, \dots, k - 1\}$  in such a way that no edge is monochromatic. The *approximate graph colouring problem* asks, given a  $k$ -colourable graph  $G$ , to find an  $\ell$ -colouring of  $G$ , where  $\ell \geq k$ . For  $k = 3$ , the state-of-the-art results are NP-hardness of the case  $\ell = 5$  [2] and a polynomial-time algorithm for finding a colouring with  $\ell = O(n^{0.19747})$  colours, where  $n$  is the number of vertices of the input graph  $G$  [9]. For non-monochromatic



© Johan Håstad, Björn Martinsson, Tamio-Vesa Nakajima, and Stanislav Živný; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 7; pp. 7:1–7:6



Leibniz International Proceedings in Informatics

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

colourings of hypergraphs, it is known that finding an  $\ell$ -colouring of a  $k$ -colourable  $r$ -uniform hypergraph is NP-hard for any constant  $\ell \geq k \geq 2$  and  $r \geq 3$  [7], and also some positive results are known for colourings with super-constantly many colours, e.g. [11, 10, 6].

A new variant of hypergraph colourings was identified in [1]. Given a 3-uniform hypergraph  $H$ , a colouring of the vertices of  $H$  with colours from the set  $\{0, 1, \dots, k-1\}$  is called a *linearly ordered (LO)  $k$ -colouring* if every edge  $e$  of  $H$  satisfies the following: if two vertices of  $e$  have the same colour then the third colour is larger. More generally, a colouring of a hypergraph  $H$  is an LO colouring if every edge of  $H$  has a unique maximum colour. (Note that the two definitions coincide for 3-uniform hypergraphs.) Barto et al. conjectured that finding an LO  $\ell$ -colouring of a 3-uniform hypergraph that admits an LO  $k$ -colouring is NP-hard for every constant  $\ell \geq k \geq 2$  [1] but even the case  $k = 2$  and  $\ell = 3$  is open. Nakajima and Živný established NP-hardness for some regimes of the parameters  $k, \ell, r$  [13, 14] and, very recently, Filakovský et al. [8] showed NP-hardness of the case  $k = 3, \ell = 4, r = 3$ . More importantly for this paper, Nakajima and Živný also considered finding an LO  $f(n)$ -colouring of an LO 2-colourable 3-uniform hypergraph with  $n$  vertices and presented polynomial-time algorithms with  $f(n) = O(\sqrt{n \log \log n} / \log n)$  [13] and  $f(n) = O(\sqrt[3]{n \log \log n} / \log n)$  [14]. Very recently, Louis, Newman, and Ray [12] have given a polynomial-time SDP-based algorithm with  $f(n) = O^*(\sqrt[5]{n})$  colours.

As our main result, we improve their results by an exponential factor.

► **Theorem 1.** *There is an algorithm which, if given a 3-uniform hypergraph  $H$  with  $n \geq 4$  vertices and  $m$  edges that admits an LO 2-colouring, finds an LO  $\log_2(n)$ -colouring of  $H$  in time  $O(n^3 + nm)$ .*

In fact we present two different algorithms that return colourings using  $O(\log n)$  colours. Both are based on solving the natural system of linear equations implied by the existence of an LO 2-colouring. In one case, the system is solved modulo 2, and in the other case, the system is solved over the rationals.

While the  $H$  which we are given as input is 3-uniform, we will need the notion that follows in greater generality; hence we define it for general hypergraphs. For each edge  $\{x_1, \dots, x_r\}$  of  $H$ , we write an equation  $v_{x_1} + \dots + v_{x_r} = 1$  where we initially use equality modulo 2 but as stated above we later use the same system over the rational numbers. Let  $A$  be this set of equations, written as a matrix with  $m$  rows and  $n$  columns. (Note that  $A$  is the *incidence matrix* of  $H$ .) Thus  $v$  is a solution if and only if  $Av = 1^m$ . Clearly a valid LO 2-colouring gives one solution but in the general case, the system has a large dimensional affine space as its set of solutions and the desired solution is hard to find.

## 2 Algorithm based on equations modulo 2

In this section all linear equations are taken modulo 2. For the following, given a set  $S$  of positive integers, an  $S$ -uniform hypergraph is a hypergraph where all edges have sizes taken from  $S$ . We first prove the following subprocedure of the main algorithm.

► **Lemma 2.** *There is an algorithm which, if given a  $\{2, 3\}$ -uniform hypergraph  $H$  with  $n$  vertices and  $m$  edges that admits an LO 2-colouring and such that the implied linear system of equations  $Av = 1^m$  does not fix the value of any variable, outputs a subset  $T$  of vertices that intersects edges of size three in zero or two vertices and edges of size two in exactly one vertex. Moreover, we have  $|T| \geq n/2$ . The algorithm runs in  $O(n^3 + nm)$  time.*

**Proof.** We first describe a randomised version of our algorithm, and then derandomise it. The set of solutions to  $Av = 1^m$  is an affine space and hence a generic solution can be written as  $v = v^0 + \sum_{i=1}^r a_i v^i$  for a basic solution  $v^0$ , linearly independent solutions to the homogeneous system  $v^i$ , and field elements (in this case bits)  $a_i$ . The fact that no variable is fixed implies that for each vertex  $x$  there is some positive  $i$  such that  $v_x^i = 1$ .

For the randomised algorithm choose  $a_1, \dots, a_r$  to be independent identically distributed uniformly random bits, and set  $T$  to be the set of variables  $x$ , such that  $v_x = 0$ . Clearly  $T$  satisfies the conclusion of the lemma as in each edge we have an odd number of ones. Since for every vertex  $x$  there exists positive  $i$  such that  $v_x^i$ , due to the influence of  $a_i v_x^i$  we see that  $x$  is included in  $T$  with probability  $1/2$ . Thus, on average  $T$  contains half the vertices.

Now, we derandomise this algorithm using the method of conditional expectations. Go through the variables  $a_i$  in increasing order and fix its value once and for all. Fixing the value of  $a_i$  determines the value of some  $v_x$  while other values remain undetermined. For each value being determined  $v_x^i = 1$  and hence one value of  $a_i$  gives the final value 0 and the other gives final value 1. Set  $a_i$  such that at least half the determined values are 0. After we have fixed all  $a_i$  this way, we have a final solution with at least  $n/2$  zeroes.

The bottleneck of the running time of this algorithm is solving the linear system of equations. This can be done in the advertised running time since every equation has  $O(1)$  entries. ◀

**Proof of Theorem 1.** As a preliminary step, we eliminate any variable determined by the system  $Av = 1^m$ . Note that if the colour of a vertex is determined by the system  $Av = 1^m$ , then this vertex must have that same colour in *all* LO 2-colourings. Fix these variables once and for all and eliminate them from the equation system. For all vertices that have been given the colour 1, we set the colours of the two other vertices in all of its edges to be 0. This process of identifying fixed variables and eliminating them is then repeated until the system  $Av = 1^m$  contains no variables fixed to a constant. At any fixed point of this process, for every edge, either all vertices in that edge are fixed (and the edge has a unique maximum as required), or exactly one vertex in it is fixed to 0.

Now, remove all coloured vertices from the hypergraph  $H$ , shrinking the edges they belonged to. The remaining hypergraph will no longer be 3-uniform, but importantly it will still be LO 2-colourable. Our goal is still to LO colour the remaining hypergraph, since any edge partially coloured by the preliminary step above must have had exactly one vertex  $v$  fixed to 0; and hence, if we LO colour the edge that resulted from removing  $v$ , this leads to an LO colouring of the original hypergraph when  $v$  is assigned 0.

Consider the following algorithm, where  $i$  starts at 0.

1. If the hypergraph  $H$  has at most, say, 20 vertices, find an LO-colouring of  $H$  by brute force using colours  $i$  and  $i + 1$ . (It exists since  $H$  is LO 2-colourable.)
2. Otherwise, find the subset  $T$  guaranteed by Lemma 2.
3. Colour the vertices in  $T$  by colour  $i$ . Remove the vertices in  $T$  from  $H$ . Remove all edges that intersect  $T$  from  $H$ . Increment  $i$  by 1.
4. Repeat.

Note that  $|T| \geq n/2$  and thus within  $-4 + \log_2 n$  repetitions we reach the first case. Each step adds one colour and we get two additional colours from the final brute-force colouring for a total of at most  $\log_2 n$  colours. The output is correct as the first time some vertex in an edge is coloured, for edges with three vertices exactly one more vertex in the same edge is coloured at the same time, and for edges with two vertices only that vertex is coloured at that time. The remaining vertex is given a higher colour and hence the edge is correctly coloured. For the time complexity, we again note that it is dominated by the time needed to solve the linear system of equations. ◀

## 7:4 A Logarithmic Approximation of Linearly-Ordered Colourings

Note that the number 20 selected above can be increased to any number that is  $O(\log n)$  and the algorithm remains polynomial time (since we must compute the colouring for a subgraph of this size by brute force). If we stop the algorithm at  $B$  vertices, then we save  $\log B + \Theta(1)$  colours, since this is how many colours the algorithm would have used to colour the last  $B$  vertices. By setting  $B = \Theta(\log n)$ , we can thus save  $\Theta(1) + \log \log n$  colours while keeping run time of the algorithm polynomial in  $n$ .

A slight variant can be obtained by instead counting the number of remaining edges with no coloured vertex. Once we have no more such vertices, we colour all remaining vertices with the next colour. For such edge, a random solution  $v$  gives the four sets of values  $(0, 0, 1)$ ,  $(0, 1, 0)$ ,  $(1, 0, 0)$  and  $(1, 1, 1)$  with equal probabilities. Thus the number of edges decreases, on average, by a factor 4 for each iteration. (Note that all the edges of size 2 are solved in the first iteration, so there is no need to count them.) It is easy to achieve this deterministically by conditional expectations. We state the conclusion as a theorem.

► **Theorem 3.** *There is an algorithm which, if given a 3-uniform hypergraph  $H$  with  $n$  vertices and  $m \geq 1$  edges that admits an LO 2-colouring, finds an LO  $(2 + \frac{1}{2} \log_2(m))$ -colouring of  $H$  in time  $O(n^3 + nm)$ .*

► **Remark 4.** Our algorithm has some similarity with algorithms for *temporal* CSPs [3]. Note that an LO  $\omega$ -colouring (which means an LO-colouring, but with no restriction on the number of colours) is a temporal CSP; to solve it, one finds a subset that could be the smallest colour (by solving mod-2 equations as above), sets that colour, then continues recursively. The difference is that for an LO  $\omega$ -colouring one does not care about the number of colours, so one can find any nonempty set of vertices to set the lowest colour to, whereas in our problem we are trying to find a large set of this kind. We note that the algorithm of [14] also uses this approach when setting “small colours”.

► **Remark 5.** We remark that the subprocedure of our algorithm computes the exclusive or of two vectors of bits. Thus the algorithm runs very fast in practice – on most architectures hundreds of operations of this kind are done at one time by (i) packing the bits within a larger word and (ii) using SIMD instructions.

### 3 Algorithm using $\mathbb{Q}$

In this section we present a more complicated algorithm which uses more colours. This might seem pointless, and indeed it might be. On the other hand the ideas used are slightly different and hence there might be situations where the ideas of this section can turn out to be useful. It is also curious to see that we can use the same system of linear equations, now over the rationals, in a rather different way. The algorithm here is in fact essentially saying that we can always use the unbalanced case of [12]. As this eliminates many complications and in particular makes it possible to completely avoid any semi-definite programming, we state all facts needed in the current section rather than refer to the very similar statements in [12]. As already stated, all arithmetic in this section is over the rational numbers. In this situation, no variables can be determined as we can set  $v^0$  to have all coordinates equal to  $1/3$ .

We study the homogeneous system  $Av = 0$  and by the assumption of LO 2-colourability it has a solution,  $w$ , with coordinates either  $-\frac{1}{3}$  or  $\frac{2}{3}$ . Let us first show how solutions over the rational numbers can be used to find LO-colourings. This is the same lemma used in the unbalanced case of [12].

► **Lemma 6.** *Suppose we have a solution,  $u$ , to the homogeneous system where  $M$  is the maximal value of the absolute value of a coordinate and  $m > 0$  is the minimal absolute value. Then we can LO-colour with  $2 + \log_2(M/m)$  colours.*

**Proof.** For notational convenience let us instead require that the minimal colour in each edge should be unique. We can simply reverse the order of the colours at the end. By scaling we can assume  $M = 1$ . We use even colours for positive coordinates and we give  $x$  the colour  $2\ell$  if  $v_x$  is at most  $2^{-(2\ell-1)}$  and strictly larger than  $2^{-(2\ell+1)}$ . For negative coordinates we use  $2\ell + 1$  as the colour if  $v_x$  is between  $-2^{-2\ell}$  (inclusive) and  $-2^{-(2\ell+2)}$  (non-inclusive). Let us verify that this gives a correct colouring.

Take an edge  $(x, y, z)$  and suppose both  $x$  and  $y$  get the same colour  $2\ell$ . Then by the linear equation of the edge  $v_z < -2^{-2\ell}$  and thus  $z$  has a colour below  $2\ell$ . The case of two vertices of odd colour is similar and as the bound on the number of colours is immediate, the lemma follows. ◀

Let us choose the vectors  $(v^i)_{i=1}^r$  giving the solutions to the homogeneous system to be of unit length and orthogonal. Define  $u = \sum_{i=1}^r y_i v^i$  where  $y_i$  are independent normal variables with mean 0 and standard deviation 1. The length of  $u$  is very close to  $\sqrt{r}$  but to be crude we use that  $E[\|u\|^2] = r$  and hence with probability  $\frac{3}{4}$  the length of  $u$  is at most  $2\sqrt{r}$ .

Let  $c^j = (v_j^1, v_j^2, \dots, v_j^r)$  be the vector of dimension  $r$  given by the  $j$ th coordinates of each vector  $v^i$ . Using this notation we see that the  $j$ th coordinate of  $u$  is a normal variable with standard deviation  $\|c^j\|$ . Recall that  $w$  is our assumed solution to  $Av = 0$  with all coordinates either  $-\frac{1}{3}$  or  $\frac{2}{3}$ . We can write  $w = \sum_{i=1}^r a_i v^i$  for some numbers  $a_i$ , and by orthonormality  $\|a\| = \|w\|$  which is at most  $\frac{2}{3}\sqrt{n}$ . As

$$|w_j| = |(c^j, a)| \leq \|c^j\| \|a\|,$$

using that  $|w_j|$  is at least  $\frac{1}{3}$  we conclude that

$$\|c^j\| \geq 1/(2\sqrt{n}).$$

The probability that a normal variable with standard deviation  $\sigma$  is of absolute value at most  $\delta$  is at most  $2\delta/(\sqrt{2\pi}\sigma)$ . We conclude that for a suitable constant  $d$  the probability that  $|u_j|$  is below  $dn^{-3/2}$  is at most  $1/(2n)$ . Thus with probability at least  $1/2$  the absolute value of any coordinate of  $u$  is at least  $dn^{-3/2}$ . Thus with probability at least  $\frac{1}{4}$ , we can apply Lemma 6 with  $M = 2n^{1/2}$  and  $m = dn^{-3/2}$  and we conclude.

► **Theorem 7.** *Using the system of linear equations over the rational numbers we can find, with probability at least  $\frac{1}{4}$  and in polynomial time, an LO-colouring with  $O(1) + 2\log_2 n$  colours.*

This algorithm is less efficient compared to the algorithm of the previous section. The main computational cost is still solving a linear system but this is more complicated over the rational numbers as coefficients are likely to grow. Our bound for the number of colours is also worse. Heuristically one could hope to have the ratio of the smallest and largest coordinate of  $u$  to be  $\Theta(n)$  but not better. Thus it is possible that we could eliminate the multiplicative constant 2 in the theorem but to get substantially fewer than  $\log n$  colours by this method sounds unlikely.

As a final observation in this section let us note that defining a colouring by the sign of the vector  $u$  we get a standard (non-monochromatic) 2-colouring of the hypergraph. This gives an alternative algorithm to that of [5, 4].

#### 4 Concluding remarks

Our algorithms indicate that LO 2-colouring is quite different from many other colouring problems. The key property that we use in our algorithm is that the constraint implies a linear constraint. The analysis of the algorithms also heavily relies on the fact that we study 3-uniform hypergraphs.

It is tempting to think that the proposed methods would extend to other constraint satisfaction problems where we are guaranteed that a solution must satisfy a linear constraint. We have so far been unable to find an interesting such example.

---

#### References

- 1 Libor Barto, Diego Battistelli, and Kevin M. Berg. Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case. In *Proc. 38th International Symposium on Theoretical Aspects of Computer Science (STACS'21)*, volume 187 of *LIPICs*, pages 10:1–10:16, 2021. doi:10.4230/LIPICs.STACS.2021.10.
- 2 Libor Barto, Jakub Bulín, Andrei A. Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. *J. ACM*, 68(4):28:1–28:66, 2021. doi:10.1145/3457606.
- 3 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *J. ACM*, 57(2):9:1–9:41, 2010. doi:10.1145/1667053.1667058.
- 4 Joshua Brakensiek and Venkatesan Guruswami. An algorithmic blend of LPs and ring equations for promise CSPs. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 436–455, 2019. doi:10.1137/1.9781611975482.28.
- 5 Joshua Brakensiek and Venkatesan Guruswami. Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy. *SIAM J. Comput.*, 50(6):1663–1700, 2021. doi:10.1137/19M128212X.
- 6 Eden Chlamtac and Gyanit Singh. Improved approximation guarantees through higher levels of SDP hierarchies. In *Proc. 11th International Workshop on Approximation, Randomization and Combinatorial Optimization (APPROX'08)*, volume 5171 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2008. doi:10.1007/978-3-540-85363-3\_5.
- 7 Irit Dinur, Oded Regev, and Clifford Smyth. The hardness of 3-uniform hypergraph coloring. *Comb.*, 25(5):519–535, September 2005. doi:10.1007/s00493-005-0032-4.
- 8 Marek Filakovský, Tamio-Vesa Nakajima, Jakub Opršal, Gianluca Tassinato, and Uli Wagner. Hardness of Linearly Ordered 4-Colouring of 3-Colourable 3-Uniform Hypergraphs. In *Proc. 41st International Symposium on Theoretical Aspects of Computer Science (STACS'24)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:19, 2024. doi:10.4230/LIPICs.STACS.2024.34.
- 9 Ken-ichi Kawarabayashi, Mikkel Thorup, and Hirotaka Yoneda. Better coloring of 3-Colorable graphs. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 331–339, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649768.
- 10 Michael Krivelevich, Ram Nathaniel, and Benny Sudakov. Approximating coloring and maximum independent sets in 3-uniform hypergraphs. *J. Algorithms*, 41(1):99–113, 2001. doi:10.1006/jagm.2001.1173.
- 11 Michael Krivelevich and Benny Sudakov. Approximate coloring of uniform hypergraphs. *J. Algorithms*, 49(1):2–12, 2003. doi:10.1016/S0196-6774(03)00077-4.
- 12 Anand Louis, Alantha Newman, and Arka Ray. Improved linearly ordered colorings of hypergraphs via SDP rounding, 2024. doi:10.48550/arXiv.2405.00427.
- 13 Tamio-Vesa Nakajima and Stanislav Živný. Linearly Ordered Colourings of Hypergraphs. In *Proc. 49th International Colloquium on Automata, Languages, and Programming (ICALP'22)*, volume 229 of *LIPICs*, pages 128:1–128:18, 2022. doi:10.4230/LIPICs.ICALP.2022.128.
- 14 Tamio-Vesa Nakajima and Stanislav Živný. Linearly Ordered Colourings of Hypergraphs. *ACM Trans. Comput. Theory*, 13(3–4), 2023. doi:10.1145/3570909.



# Speed-Robust Scheduling Revisited

Josef Minařík ✉

Computer Science Institute of Charles Univ., Faculty of Mathematics and Physics, Prague, Czechia

Jiří Sgall ✉ 

Computer Science Institute of Charles Univ., Faculty of Mathematics and Physics, Prague, Czechia

---

## Abstract

Speed-robust scheduling is the following two-stage problem of scheduling  $n$  jobs on  $m$  uniformly related machines. In the first stage, the algorithm receives the value of  $m$  and the processing times of  $n$  jobs; it has to partition the jobs into  $b$  groups called bags. In the second stage, the machine speeds are revealed and the bags are assigned to the machines, i.e., the algorithm produces a schedule where all the jobs in the same bag are assigned to the same machine. The objective is to minimize the makespan (the length of the schedule). The algorithm is compared to the optimal schedule and it is called  $\rho$ -robust, if its makespan is always at most  $\rho$  times the optimal one.

Our main result is an improved bound for equal-size jobs for  $b = m$ . We give an upper bound of 1.6. This improves previous bound of 1.8 and it is almost tight in the light of previous lower bound of 1.58. Second, for infinitesimally small jobs, we give tight upper and lower bounds for the case when  $b \geq m$ . This generalizes and simplifies the previous bounds for  $b = m$ . Finally, we introduce a new special case with relatively small jobs for which we give an algorithm whose robustness is close to that of infinitesimal jobs and thus gives better than 2-robust for a large class of inputs.

**2012 ACM Subject Classification** Theory of computation → Online algorithms; Theory of computation → Scheduling algorithms

**Keywords and phrases** scheduling, approximation algorithms, makespan, uniform speeds

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.8

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.11670> [8]

**Funding** Partially supported by grant 24-10306S of GA ČR.

**Acknowledgements** We are grateful to Franziska Eberle for insightful discussions and comments, and to the anonymous referees for detailed and useful reviews.

## 1 Introduction

Speed-robust scheduling is a two-stage problem that was introduced by Eberle *et al.* [4]. The eventual goal is to schedule on  $m$  uniformly related machines, however their speeds are not known at the beginning. In the first stage, the algorithm receives the value of  $m$  and the processing times of  $n$  jobs; it has to partition the jobs into  $b$  groups called bags. In the second stage, the machine speeds are revealed and the bags are assigned to the machines, i.e., the algorithm produces a schedule where all the jobs in the same bag are assigned to the same machine. The objective is to minimize the makespan (the length of the schedule). The algorithm is compared to the optimal schedule of the jobs on the machines with known speeds; it is called  $\rho$ -robust, if its makespan is always at most  $\rho$  times the optimal one.

This problem is motivated by situations like the following one. Suppose that you have  $n$  computational tasks that you want to solve. You have a computational cluster available, but with unknown parameters. You only know that there will be (at most)  $m$  machines available on the cluster. You do not know anything about the performance of the machines – some of the machines might be faster than others; you only know that there will be (at most)  $m$



© Josef Minařík and Jiří Sgall;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 8; pp. 8:1–8:20



Leibniz International Proceedings in Informatics

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

machines available on the cluster. Furthermore, you can submit at most  $b$  different tasks to the cluster. Hence you will have to partition your  $n$  tasks into at most  $b$  groups. One such group will then have to be executed on one machine. The cluster will then schedule the groups optimally, knowing the speeds of the machines, and minimize the makespan.

Studying uncertainty in scheduling has a long history. In the classical online scheduling [10], the machine environment is usually fixed and the uncertainty stems from job arrivals. Considering uncertainty in the machine environment is less frequent. One early example is the work of Csanád and Noga [7], where additional machines can be bought for a certain cost. A substantial body of research with changing machine speeds is the area of dynamic speed scaling, in particular in the context of minimizing the power consumption, see [1, 9]; however, note that here the changing speeds are not a part of the adversarial environment but used by the algorithm to its advantage. Another direction considers online scheduling with unavailability periods [3]. One-machine scheduling with adversarially changing machine speed was considered in [5] in the context of unreliable machines.

Completely reversing the scenario with all jobs known from the beginning but uncertain machine environment is a recent new model introduced by Stein and Zhong [11] and Eberle et al. [4], see Section 1.2.

### 1.1 Formal definitions

Formally, in the first stage, we receive three positive integers  $n, m, b$  and  $n$  non-negative real numbers  $p_1, \dots, p_n$  representing the processing times of  $n$  jobs. The total processing time is denoted  $P = \sum_{j=1}^n p_j$ . The output of our first-stage algorithm is a mapping  $B: \{1, \dots, n\} \rightarrow \{1, \dots, b\}$ , where  $B(j) = i$  represents the fact that the job  $j$  was assigned to the bag  $i$ . The sum of the processing times of all the jobs assigned to bag  $i$  the *size* of bag  $i$  and denoted  $a_i = \sum_{j: B(j)=i} p_j$ . The exact mapping  $B$  is not important for the second stage since the makespan depends only on the bag sizes.

In the second stage, we are given the bag sizes  $a_1, \dots, a_b$  and the previously unknown machine speeds  $s_1, \dots, s_m \geq 0$ , not all equal to 0. We partition the bag indices  $\{1, \dots, b\}$  into  $m$  sets  $M_1, \dots, M_m$ , representing the assignment to the  $m$  machines. Machine  $i$  then has a completion time  $C_i = (\sum_{j \in M_i} a_j) / s_i$ ; for  $s_i = 0$  we require  $M_i = \emptyset$  and set  $C_i = 0$ , i.e., machine of speed 0 does not accept any jobs. Finally,  $C_{\max} = \max_{i=1}^m C_i$  is the makespan, i.e., the length of the schedule.

Let  $C_{\max}^*$  denote the makespan of the adversary, who does not have to create bags and can assign jobs directly to machines. Alternatively and equivalently, the adversary also creates bags, but with the knowledge of the speeds already in the first stage.

We call a first stage algorithm  $\rho$ -robust if, for all possible inputs and for all possible choices of machine speeds, there exists a second-stage assignment of bags to machines such that  $C_{\max} \leq \rho \cdot C_{\max}^*$ . Intuitively, an algorithm is  $\rho$ -robust if it performs at most  $\rho$  times worse than the adversary.

The previous definition implicitly assumes that the second stage is solved optimally. This is reasonable, as the scheduling on uniformly related machines allows PTAS, see [6, 12], so the chosen (presumably optimal) second-stage solution can be replaced by an arbitrarily good approximation. Also, our proofs show that the second-stage algorithm can be implemented by efficient greedy algorithms without any loss of performance, once the optimal makespan or its approximation is known.

We call the special cases of the problem *sand*, *bricks*, *rocks* and *pebbles*. Sand, bricks, and rocks were introduced by Eberle et al. [4]. These words represent the types of jobs.

- Rocks can be any shape or size and represent jobs of arbitrary processing time. This is the most general setting.
- Bricks are all the same and represent jobs with equal processing times.
- Sand grains are very small and represent infinitesimally small processing times.
- Pebbles represent jobs that are relatively small compared to the average load of all machines. We call an instance of speed-robust scheduling  $q$ -pebbles if  $p_j \leq q \cdot \frac{P}{m}$  holds for all jobs  $j$ .

## 1.2 Previous results

The two-stage scheduling problem with uncertainty in the machine environment was introduced by Stein and Zhong [11]. They focused on the case of  $m$  identical machines where in the second stage some machine might fail and then do not process any tasks. This amounts to a special case of speed-robust scheduling where  $s_i \in \{0, 1\}$  for  $1 \leq i \leq m$ . They gave lower bounds of  $4/3$  for equal-size jobs (bricks) and  $(\sqrt{2} + 1)/2 \approx 1.207$  for infinitesimal jobs (sand). Their algorithms were later improved by Eberle *et al.* [4] to algorithms matching the lower bounds in both cases.

Our immediate predecessor, Eberle *et al.* [4], introduced the speed-robust scheduling for general speeds, i.e., on uniformly related machines. They studied mainly the case  $b = m$ , i.e., the case when the number of bags is equal to the number of machines. For this case they gave tight bounds for sand for every  $m$ , for large  $m$  the bound approaches  $e/(e - 1) \approx 1.58$ . For equal-size jobs (bricks), they have shown an upper bound of 1.8.

For the most general case of rocks, the strongest known result is the algorithm with the robustness factor at most  $1 + (m - 1)/b$ , which equals  $2 - 1/m$  for  $b = m$ , given also by Eberle *et al.* [4]. It remains an interesting open problem to improve this bound, in particular to give an upper bound  $2 - \varepsilon$  for rocks and  $b = m$ .

## 1.3 Our results

We now describe our results and compare them to the previous ones in each of the scenarios.

**Sand.** For sand, we give matching lower and upper bounds for any  $b$  and  $m$ . Namely, for  $b \geq m$  we give an optimal algorithm which is  $\bar{\rho}(m, b)$ -robust for

$$\bar{\rho}(m, b) = \frac{m^b}{m^b - (m - 1)^b} = \frac{1}{1 - \left(1 - \frac{1}{m}\right)^b}. \quad (1.1)$$

This matches the results of Eberle *et al.* [4] who gave an algorithm with the robustness factor equal to  $\bar{\rho}(m, b) \leq e/(e - 1) \approx 1.58$  for  $b = m$ , generalizes them to arbitrary  $b \geq m$  and significantly simplifies the proof.

An interesting case is when the number of bags is a constant multiple of  $m$ . For a fixed  $\alpha \geq 1$  and  $b = \alpha m$ , our bound approaches  $1/(1 - e^{-\alpha})$  from below for a large  $m$ . For example, doubling the number of bags to  $b = 2m$  decreases the robustness factor from 1.58 to 1.16.

If  $b < m$ , the second-stage algorithm uses only the  $b$  fastest machines, so we can decrease  $m$  to  $m' = b$  and tight results with robustness factor  $\bar{\rho}(m', b) = \bar{\rho}(b, b)$  follow already from [4].

**Pebbles.** For the new case of  $q$ -pebbles and  $b \geq m$ , we give a  $(\bar{\rho}(m, b) + q)$ -robust algorithm. For  $p < 0.42$ , this gives an algorithm with the robustness factor below 2, i.e., below the currently strongest known upper bound for rocks.

**Bricks.** As our main result, we give a 1.6-robust algorithm for bricks for  $b = m$ . This improves the bound of 1.8 from Eberle *et al.* [4].

Furthermore, as a direct application of our results for pebbles we give a  $(\bar{\rho}(m, b) + m/n)$ -robust algorithm for any  $n$  and  $b \geq m$ . This improves and generalizes the  $((1 + m/n)\bar{\rho}(m, m))$ -robust algorithm for  $b = m$  given by Eberle *et al.* [4]. Namely, we improve the multiplicative factor of  $(1 + m/n)$  to only an additive term of  $m/n$ .

**Structure of the paper.** We give some general preliminaries in Section 2. We give the results for sand and pebbles in Sections 3 and 4. We focus on our main result for bricks in Section 5. Some small cases need computer verification or tabulation of parameters, results of these are given in the full version of the paper on arXiv [8].

## 2 Preliminaries

We assume that the processing times, the machine speeds, and the bag sizes are always listed in a non-increasing order.

In the rest of this paper, we will make two assumptions below that restrict the speeds to particular special cases. This is without loss of generality, leveraging the fact that the algorithm must commit the bag sizes in the first phase without knowing the speeds.

- The optimal makespan is equal to 1. This implies that the robustness factor is equal to the makespan of the algorithm.

Scaling all the speeds does not change the ratio of the makespans of our algorithm and the adversary. Thus for every instance of the problem, there exists another instance with  $C_{\max}^* = 1$  that differs only in the speeds and the ratio of makespans of our algorithm and the adversary remains the same. It follows that any first-stage algorithm that is  $\rho$ -robust for instances with  $C_{\max}^* = 1$  is  $\rho$ -robust for general instances, too.

- The sum of the processing times of all jobs equals to the sum of the speeds of all the machines, i.e.,  $P = \sum_{i=1}^m p_i = \sum_{i=1}^m s_i$ . In other words, the adversary is fully utilizing all the machines, and the completion time of all the machines with non-zero speed is equal to 1, using the previous assumption.

If there is some machine  $i$  with  $s_i > 0$  and completion time  $C_i^* < 1$  in the optimal schedule, we change its speed to  $s'_i = C_i^* s_i$ . This does not change the optimal makespan of the adversary and the makespan of the algorithm can only increase. Once again, it follows that any first-stage algorithm that is  $\rho$ -robust for these special instances is also  $\rho$ -robust for general instances.

For the second stage, typically, we use a simple greedy algorithm for the second stage instead of analyzing the optimal schedule. Technically, for an algorithm we need to know the optimal makespan (to modify the speeds appropriately, according to the assumptions above). However note that first we can approximate the makespan and second the algorithm is only used as a tool in the analysis.

For sand and pebbles we use Algorithm GREEDYASSIGNMENT (see below), a variant of the well-known LPT algorithm. It is parameterized by  $\rho$ , the robustness factor to be achieved. At the beginning, every machine is assigned a capacity equal to its speed multiplied by  $\rho$ . The algorithm then goes through all the bags from large to small, assigns them on the machine with the largest capacity remaining, and decreases the capacity appropriately. If the capacities remain non-negative at the end, the makespan of the created assignment is at most  $\rho$  since machine  $i$  has been assigned jobs of total processing times at most  $\rho s_i$ .

We use this to formulate the following sufficient condition for  $\rho$ -robustness of an algorithm which is instrumental in proving the upper bounds for sand and pebbles.

■ **Algorithm** GREEDYASSIGNMENT.

---

**Input:** bag sizes  $a_1 \geq \dots \geq a_b$ ; machine speeds  $s_1 \geq \dots \geq s_m$ ; desired robustness factor  $\rho$

**for**  $i \leftarrow 1$  **to**  $m$  **do**

$c_i \leftarrow \rho s_i$  ▷ Initialize the capacities of all machines

$M_i = \emptyset$  ▷ Initialize the assignment

**for**  $k \leftarrow 1$  **to**  $b$  **do**

$i \leftarrow$  index of a machine with the largest  $c_i$

$M_i \leftarrow M_i \cup \{k\}$  ▷ Assign bag  $k$  to machine  $i$

$c_i \leftarrow c_i - a_k$  ▷ Decrease the remaining capacity of the selected machine

**return**  $M_1, \dots, M_m$

---

▶ **Theorem 2.1.** *If a first-stage algorithm always produces bag sizes satisfying inequalities  $a_k \leq \frac{\rho P - \sum_{j=1}^{k-1} a_j}{m}$ , for all  $k = 1, \dots, b$ , then the algorithm is  $\rho$ -robust.*

**Proof.** Recall that we assume  $\sum_{i=1}^m s_i = P$ . We claim that the second-stage algorithm GREEDYASSIGNMENT produces an assignment with makespan at most  $\rho$ . We only need to show that there is a machine with capacity at least  $a_k$  when assigning the  $k$ th bag. The initial total capacity was  $\rho P$  and was already decreased by  $\sum_{j=1}^{k-1} a_j$  at the time of assigning bag  $a_k$ . It follows that the remaining capacity is equal to  $\rho P - \sum_{j=1}^{k-1} a_j$  and thus there exists a machine with capacity at least  $(\rho P - \sum_{j=1}^{k-1} a_j)/m \geq a_k$ . ◀

**3 Sand**

Intuitively, the case of sand corresponds to the limit case where  $n$  is large and all the jobs are small and have equal sizes. One can view this as an infinite number of infinitesimal jobs.

More formally, we are given just  $m, b$ , and  $P$  as an input of the first stage. The result of the first stage are  $b$  non-negative reals  $a_1, \dots, a_b$  whose sum equals  $P$ . The formulation of the second stage remains the same.

The model of infinitesimally small jobs resembles preemptive scheduling. In the optimal algorithm for preemptive scheduling [2], one needs to maintain the loads of machines in a geometric sequence with common ratio  $m/(m-1)$  for  $m$  machines, roughly speaking. The proofs for sand show that here the same geometric sequence is also crucial, in particular it is used for the bag sizes in the algorithm. We now describe the sequence and state its properties useful both for the upper and lower bounds.

We define  $U = m^b$ ,  $L = m^b - (m-1)^b$  and  $t_j = m^{b-j}(m-1)^{j-1}$  for  $j \in \{1, \dots, b\}$ .

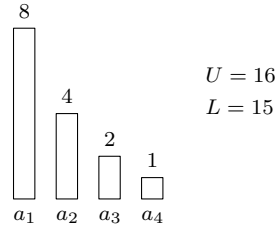
Observe that equation (1.1) defines  $\bar{\rho}$  as  $\bar{\rho}(m, b) = U/L$ .

▶ **Lemma 3.1.** *For all  $k = 1, \dots, b$ , it holds that  $\sum_{j=1}^k t_j = U - (m-1)t_k$ . In particular,  $\sum_{j=1}^b t_j = U - (m-1)^b = L$ .*

**Proof.** We proceed by induction on  $k$ . The lemma holds for  $k = 1$  since  $U = mt_1$  and thus  $t_1 = U - (m-1)t_1$ . Now suppose it holds for  $k$ . We can derive

$$\sum_{j=1}^{k+1} t_j = U - (m-1)t_k + t_{k+1} = U - m^{b-k}(m-1)^k + m^{b-k-1}(m-1)^k = U - (m-1)t_{k+1}$$

which completes the induction step. ◀



■ **Figure 3.1** An example of bag sizes chosen for  $m = 2$  and  $b = 4$ .

To get some intuition behind the algorithm for sand, it might be useful to consider the case  $m = 2$ , see Figure 3.1. Suppose that  $P = L = 2^b - 1$ , choose the bag sizes  $a_k = t_k = 2^{b-k}$ . For  $m = 2$  the sizes are powers of two, so it is easy to see that we can achieve the robustness ratio of  $1 + 1/(2^b - 1) = 2^b/(2^b - 1)$  as follows: The adversary chooses any speeds  $s_1, s_2$  such that  $s_1 + s_2 = P = 2^b - 1$ . The capacities of the machines (as in GREEDYASSIGNMENT) then satisfy  $c_1 + c_2 = 2^b$  and thus  $\lfloor c_1 \rfloor + \lfloor c_2 \rfloor \geq 2^b - 1$ . We can express  $\lfloor c_1 \rfloor$  in binary, assign the corresponding bags on the first machine and the remaining bags to the second machine.

### 3.1 Upper bound

We use a different approach than Eberle et al. [4] for the proof of the upper bound. We choose the same bag sizes (for  $b = m$ ) but we simplify the proof by use of Theorem 2.1. Algorithm SAND describes the bag sizes. Note that the sum of bag sizes produced by SAND is  $P$ , using Lemma 3.1.

■ **Algorithm** SAND.

---

**Input:** number of bags  $b$ ; number of machines  $m$ ; total amount of sand  $P$   
 $L \leftarrow m^b - (m - 1)^b$   
**for**  $j \leftarrow 1$  to  $b$  **do**  $a_j \leftarrow t_j \frac{P}{L}$   
**return**  $a_1, a_2, \dots, a_b$

---

► **Theorem 3.2.** *Algorithm SAND is  $\bar{\rho}(m, b)$ -robust for sand, for  $\bar{\rho}$  defined by (1.1).*

**Proof.** We assume  $P = L$  since it does not change the ratio of our makespan and the makespan of the adversary. Under this assumption, SAND produces bag sizes  $a_k = t_k$ .

It is sufficient to show that the bag sizes produced by SAND satisfy the condition of Theorem 2.1. Let us prove the  $k$ th inequality in the assumption of the theorem. We have

$$\bar{\rho}(m, b)P - \sum_{j=1}^{k-1} a_j = \frac{U}{L}L - \sum_{j=1}^{k-1} t_j = U - \sum_{j=1}^k t_j + t_k.$$

According to Lemma 3.1, we can simplify the right-hand side as follows.

$$U - \sum_{j=1}^k t_j + t_k = U - (U - (m - 1)t_k) + t_k = mt_k = ma_k.$$

The  $k$ th inequality in the assumption of Theorem 2.1 follows, in fact it holds with equality.

Theorem 2.1 now implies that there exists an assignment with makespan at most  $\bar{\rho}(m, b)$ . ◀

### 3.2 Lower bound

The following proof is a slightly modified and generalized version of the proof by Eberle et al. [4]. The main difference is that we do not require the number of bags and machines to be the same.

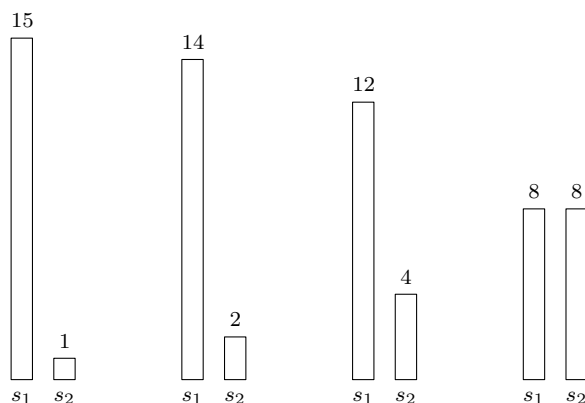
► **Theorem 3.3.** *No deterministic algorithm for sand may have a robustness factor smaller than  $\bar{\rho}(m, b)$ , for  $\bar{\rho}$  defined by (1.1).*

**Proof.** Let us without loss of generality assume  $P = U$  (be aware that we assumed  $P = L$  in the proof of the upper bound). Let us denote the chosen bag sizes by  $a_1 \geq \dots \geq a_b$ . We will restrict the adversary to  $b$  different speed configurations indexed by  $k$ , where

$$\mathcal{S}_k = \{s_1 = U - (m - 1)t_k, s_2 = t_k, s_3 = t_k, \dots, s_m = t_k\}.$$

See Figure 3.2 for an example. Note that the sum of machine speeds is equal to  $U$  in every configuration and hence the makespan of the adversary is indeed 1 as we always assume. In every speed configuration, there are  $m - 1$  slow machines and one fast machine, since

$$s_1 = U - (m - 1)t_k = \sum_{j=1}^k t_j \geq t_k.$$



■ **Figure 3.2** An example of speed configurations considered by the adversary for  $m = 2$  and  $b = 4$ .

Let  $k_{\max}$  be the largest index such that  $a_{k_{\max}} \geq \frac{U}{L}t_{k_{\max}}$ . This index must exist since

$$\sum_{j=1}^b a_j = U = \frac{U}{L}L = \frac{U}{L} \sum_{j=1}^b t_j.$$

Now let the adversary choose the speed configuration  $\mathcal{S}_{k_{\max}}$ . We distinguish two cases depending on the bag assignment in the second stage.

**Case 1.** At least one of the bags  $a_1, \dots, a_{k_{\max}}$  is assigned to a slow machine. The makespan is at least the completion time of this machine which is at least

$$\frac{a_j}{t_{k_{\max}}} \geq \frac{a_{k_{\max}}}{t_{k_{\max}}} \geq \frac{U}{L}.$$

## 8:8 Speed-Robust Scheduling Revisited

**Case 2.** All of the bags  $a_1, \dots, a_{k_{\max}}$  are assigned to the fast machine. Total size of the bags assigned to the fast machine is at least

$$\sum_{j=1}^{k_{\max}} a_j = U - \sum_{j=k_{\max}+1}^b a_j.$$

By definition of  $k_{\max}$  it holds that  $a_j < \frac{U}{L}t_j$  for  $j > k_{\max}$  and we can bound

$$U - \sum_{j=k_{\max}+1}^b a_j \geq U - \frac{U}{L} \sum_{j=k_{\max}+1}^b t_j.$$

Since  $\sum_{j=1}^b t_j = L$ , we can rearrange the right-hand side as follows

$$U - \frac{U}{L} \sum_{j=k_{\max}+1}^b t_j = U - \frac{U}{L} \left( L - \sum_{j=1}^{k_{\max}} t_j \right) = \frac{U}{L} \sum_{j=1}^{k_{\max}} t_j.$$

By Lemma 3.1 it holds that

$$\frac{U}{L} \sum_{j=1}^{k_{\max}} t_j = \frac{U}{L} (U - (m-1)t_{k_{\max}}) = \frac{U}{L} s_1$$

due to the choice of  $s_1$  in the configuration  $\mathcal{S}_{k_{\max}}$ . Thus the makespan would be at least  $U/L = \bar{\rho}(m, b)$ .

The makespan was at least  $U/L$  in both cases, hence the robustness factor is at least  $U/L = \bar{\rho}(m, b)$  and the theorem follows.  $\blacktriangleleft$

### 4 Pebbles

Recall that an instance of our problem is called  $q$ -pebbles if the processing times satisfy

$$p_j \leq q \cdot \frac{P}{m} = q \cdot \frac{\sum_{\ell=1}^n p_\ell}{m}.$$

This definition might seem a bit unnatural at the first glance, but there is a very intuitive formulation. The expression  $\frac{P}{m}$  represents the *average load of a machine*. The definition of pebbles says that the processing times are relatively small compared to the average load of all machines.

Without loss of generality we assume in this section that the sum of processing times is  $P = m$ . This transforms the condition for  $q$ -pebbles from the definition into

$$p_j \leq q,$$

which is easy to work with.

We use similar ideas as in the optimal algorithm for sand. Recall the condition of Theorem 2.1

$$a_k \leq \frac{\rho P - \sum_{j=1}^{k-1} a_j}{m}.$$

As we have already noticed in Section 3.1, the optimal bag sizes for sand not only satisfy the above inequality, they actually have equality there. The bag sizes for sand are given by the recurrence

$$a_k = \frac{\bar{\rho}(m, b)P - \sum_{j=1}^{k-1} a_j}{m}.$$



When we in addition assume  $P = m$ , as in the case of pebbles, we get

$$a_k = \bar{\rho}(m, b) - \frac{1}{m} \sum_{j=1}^{k-1} a_j. \quad (4.1)$$

Let  $a_1, \dots, a_b$  denote values given by the recurrence (4.1) for the rest of this section. Remember that the sum of  $a_1, \dots, a_b$  equals  $P$ . Let us denote the bag sizes we will be choosing for pebbles  $d_1, \dots, d_b$ . We again want to use Theorem 2.1. In other words, for the desired robustness factor  $\rho$ , we want the bag sizes to satisfy

$$d_k \leq \rho - \frac{1}{m} \sum_{j=1}^{k-1} d_j. \quad (4.2)$$

Consider the following algorithm. Place as many pebbles as you can into the first bag while it satisfies the inequality (4.2). Then do the same for the second bag and so on until the last bag (or until we run out of jobs). See PEBBLES for pseudocode.

■ **Algorithm** PEBBLES.

---

**Input:** processing times  $p_1 \geq \dots \geq p_m$ ; number of machines  $m$ ; number of bags  $b$ ; desired robustness factor  $\rho$   
 $B \leftarrow$  empty mapping  
**for**  $k \leftarrow 1$  to  $b$  **do**  $d_k \leftarrow 0$  ▷  $d_k$  represents the size of the  $k$ th bag  
 $k \leftarrow 1$  ▷  $k$  represents index of currently considered bag  
**for**  $j \leftarrow 1$  to  $n$  **do**  
    **while**  $k \leq b$  **and**  $d_k + p_j > \rho - \frac{1}{m} \sum_{\ell=1}^{k-1} d_\ell$  **do**  $k \leftarrow k + 1$   
    **if**  $k > b$  **then break**  
     $B[j] \leftarrow k$   
     $d_k \leftarrow d_k + p_j$   
**return**  $B$

---

► **Theorem 4.1.** *There exists a  $(\bar{\rho}(m, b) + q)$ -robust algorithm for  $q$ -pebbles, for  $\bar{\rho}$  defined by (1.1).*

**Proof.** We show that Algorithm PEBBLES puts every job in some bag for  $\rho = \bar{\rho}(m, b) + q$ .

Suppose for a contradiction that the algorithm does not use all the jobs. Then the bag sizes  $d_k$  at the end of the algorithm must satisfy

$$d_k + q > \rho - \frac{1}{m} \sum_{j=1}^{k-1} d_j.$$

Indeed, if for some  $k$  this inequality is not satisfied, adding one more job of size at most  $p$  to bag  $k$  would not violate the inequality (4.2) and the algorithm would have done so.

Plugging in the expression for  $\rho$  gives us

$$d_k > \bar{\rho}(m, b) - \frac{1}{m} \sum_{j=1}^{k-1} d_j. \quad (4.3)$$

We are going to show

$$\sum_{j=1}^k d_j \geq \sum_{j=1}^k a_j, \quad (4.4)$$

## 8:10 Speed-Robust Scheduling Revisited

for all  $k \in \{0, \dots, b\}$ . We prove this claim by induction. The case  $k = 0$  is trivial since the summations are empty and both sides are equal to 0. Let us now prove the induction step for  $k$  using the equation (4.1) and the inequality (4.3).

$$d_k - a_k \geq \left( \bar{\rho}(m, b) - \frac{1}{m} \sum_{j=1}^{k-1} d_j \right) - \left( \bar{\rho}(m, b) - \frac{1}{m} \sum_{j=1}^{k-1} a_j \right) = -\frac{1}{m} \left( \sum_{j=1}^{k-1} d_j - \sum_{j=1}^{k-1} a_j \right)$$

We can now easily finish the induction step. We simplify

$$\begin{aligned} \sum_{j=1}^k d_j - \sum_{j=1}^k a_j &= \left( \sum_{j=1}^{k-1} d_j - \sum_{j=1}^{k-1} a_j \right) + (d_k - a_k) \\ &\geq \left( \sum_{j=1}^{k-1} d_j - \sum_{j=1}^{k-1} a_j \right) - \frac{1}{m} \left( \sum_{j=1}^{k-1} d_j - \sum_{j=1}^{k-1} a_j \right) = \frac{m-1}{m} \left( \sum_{j=1}^{k-1} d_j - \sum_{j=1}^{k-1} a_j \right), \end{aligned}$$

which is non-negative by the induction hypothesis for  $k-1$  and thus the claim (4.4) holds.

Using the claim (4.4) for  $k = b$  gives us

$$\sum_{j=1}^b d_j \geq \sum_{j=1}^b a_j = P,$$

which is a contradiction with the assumption that we did not use all jobs.  $\blacktriangleleft$

It is interesting to take a look at the case  $b = m$ . Theorem 4.1 implies that there exists an algorithm with robustness factor at most

$$\frac{e}{e-1} + q \approx 1.58 + q.$$

The best known result for rocks gives robustness factor  $2 - 1/m$ . This gets arbitrarily close to 2 for large  $m$ . Hence we have obtained a stronger result for

$$q < 2 - \frac{e}{e-1} \approx 0.42.$$

## 5 Bricks

In this section, we study the case of jobs with equal processing times. An important parameter is the ratio of the number of jobs and the number of machines, which we denote  $\lambda = n/m$ . We can scale the instance so that  $p_j = 1$  for all  $j$ , which we assume from now on. Note that now  $P = n$  and the average load is  $P/m = n/m = \lambda$ .

Thus the instance satisfies the definition of  $p$ -pebbles for  $p = 1/\lambda$ . Theorem 4.1 immediately implies our first improved bound for bricks:

► **Theorem 5.1.** *There exists an algorithm with robustness factor at most  $\bar{\rho}(b, m) + m/n$  solving the problem for  $n$  bricks,  $m$  machines and  $b$  bags.*  $\blacktriangleleft$

In the rest of this section we focus on our main result, the 1.6-robust algorithm for bricks in case  $b = m$ . This will have the following ingredients:

- For  $\lambda \geq 60$  we have  $e/(e-1) + 1/60 < 1.6$ , so by Theorem 5.1 we can use Algorithm PEBBLES.
- For  $\lambda < 60$  we design a new algorithm BRICKS. We split the analysis into two cases.

- For  $m \geq 144$ , we modify its solution into a certain fractional solution, which is easier to analyze, and bound the difference between the two solutions.
  - For  $m < 144$ , we have a finite number of instances, which we verify using a computer.
- We stress that the analysis of instances for  $m < 144$  shows that Algorithm BRICKS works here without any changes, too, i.e., it does not lead to an algorithm with exploding number of cases tailored to specific inputs.

### 5.1 First stage algorithm BRICKS

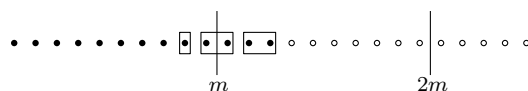
Our assumptions on the optimal solution explained at the beginning of Section 2 imply that we can also restrict ourselves to instances with  $\sum_{i=1}^m s_i = n$  and furthermore the values of speeds  $s_i$  are integral, as in the optimal solution the machine loads are necessarily integral. (Recall that this is due to the fact that we can modify the speeds independently of the first-stage algorithm.)

The key ingredient of the improved algorithm is to observe that the integrality of speeds allows us to use the pigeonhole principle to create larger bags. Furthermore, with appropriate accounting we can use the pigeonhole principle iteratively.

Let us demonstrate this on an example. Let  $n = 13$ ,  $m = 10$  and  $\rho = 1.6$ . The total speed of 10 machines is 13, so one machine has speed at least 2. This means that one of the machines will have capacity  $2\rho = 3.2$  and we can create and assign a bag of size  $\lfloor 2\rho \rfloor = 3$ . Without integrality of the speeds, only a machine with speed 1.3 would be guaranteed, so the capacity would be just above 2.

To continue iteratively, we cannot reason about the capacity as in Algorithm GREEDYASSIGNMENT. Instead, for each bag we reserve some integral amount of speed on one of the machines. For this accounting, we represent the remaining unreserved total speed by *coins*.

In the example above, we pay 2 coins for a bag of size 3. This seems like an overpayment compared to Algorithm GREEDYASSIGNMENT, as the 2 coins correspond to capacity 3.2, so we waste a capacity of 0.2. However, after this step, we are left with 11 coins among the 10 machines, and using the integrality and the pigeonhole principle once more, we are guaranteed to have one machine with 2 coins (these coins may be on a different machine or they may be the ones remaining on the same machine). Thus we can create another bag of size 3. Now there are only 9 coins remaining and we can only create a bag of size 1 at cost 1. See Figure 5.1 for an illustration. Overall, the effect of integrality is more significant than the overpayment due to rounding, and thus we are able to obtain an improved algorithm.



■ **Figure 5.1** Graphical representation of the first three chosen bags for  $n = 13$ ,  $m = 10$ . The dots represent coins and the boxes represent chosen bags. The number of coins inside a box represent the cost of the bag. Vertical lines emphasize the multiples of  $m$ , which determine the bag costs.

Formally, we start Algorithm BRICKS with  $c = n$  coins. In each round we pay  $z = \lfloor c/m \rfloor$ , create a bag of size  $\lfloor z \cdot \rho \rfloor$  and continue with remaining coins on  $m$  machines. The *cost* of a bag is the number of coins we pay for it, i.e.,  $z$  in the algorithm.

If Algorithm BRICKS produces bags of total size at least  $n$ , we say it is *successful*. If the total sum of bag sizes exceeds  $n$ , we decrease the sizes of some bags to make the sum equal to  $n$ . E.g., we can remove some of the last small bags and then decrease size of the last non-empty bag as needed.

■ **Algorithm** BRICKS.

---

**Input:** number of bricks  $n$ ; number of machines  $m$ ; number of bags  $b$ ; desired robustness factor  $\rho$

$c \leftarrow n$  ▷ The initial number of coins is  $n$

**for**  $j \leftarrow 1$  to  $b$  **do**

$z \leftarrow \lceil c/m \rceil$  ▷ max guaranteed coins on a machine

$a_j \leftarrow \lfloor z \cdot \rho \rfloor$  ▷ max integer such that  $\text{cost}(a_j) = z$

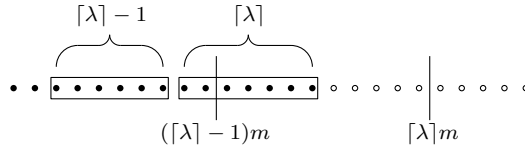
$c \leftarrow c - z$

**return**  $a_1, a_2, \dots, a_b$

---

In Section 5.3 we show that this algorithm is sound, namely, we give a modification of the second stage algorithm algorithm GREEDYASSIGNMENT for which we show that a machine with unused speed  $z$  always exists and thus we can assign all bags.

For a general instance, there is always a machine of speed at least  $\lceil n/m \rceil = \lceil \lambda \rceil$ , and thus the cost of the first bag is chosen as  $\lceil \lambda \rceil$ . The cost will then decrease by 1 every time the number of coins decreases below a multiple of  $m$ . Figure 5.2 illustrates this.



■ **Figure 5.2** Graphical representation of the first chosen bag of size  $\lceil \lambda \rceil$ .

Note that the costs of the bags chosen by BRICKS do not depend on  $\rho$ . The sizes of the bags, however, do depend on  $\rho$ . See Figure 5.3 below for an example execution of BRICKS for  $n = 45$  and  $m = 9$ . This execution shows that BRICKS fails for  $\rho < 1.6$  but succeeds for  $\rho = 1.6$ .

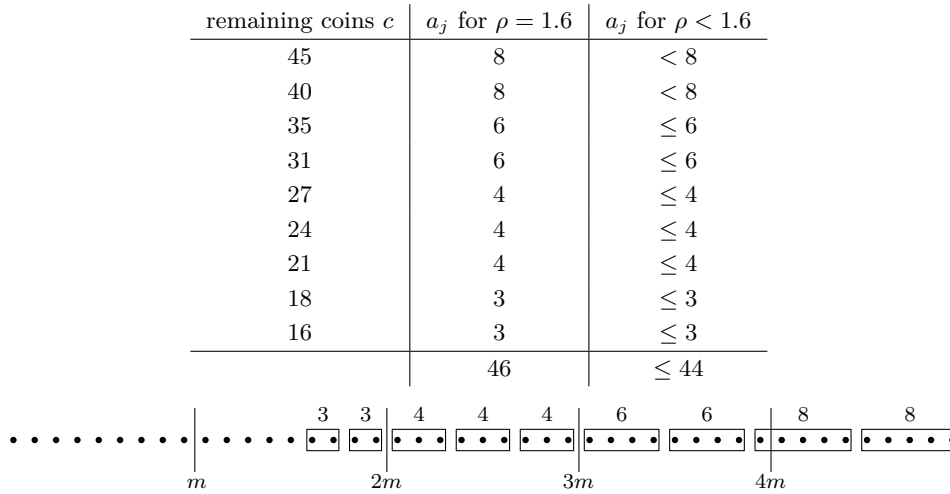
## 5.2 Fractional solutions

In general, the cost of the first bag chosen will be  $\lceil \lambda \rceil$ . The cost will then decrease by 1 every time the number of coins decreases below a multiple of  $m$ . Roughly speaking, we use approximately  $m$  coins for bags of each size.

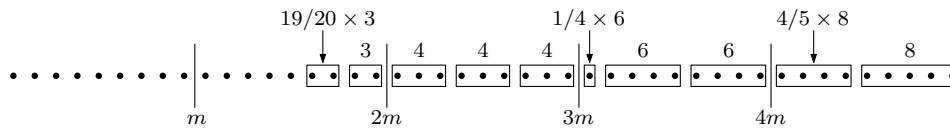
We need to show that the created  $b$  bags have total size at least  $n$ . If we would use exactly  $m/z$  bags for each cost  $z$ , the total size of bags is easy to compute. However, the integral number of bags of each cost causes rounding issues when the bag cost decreases and these complicate the calculations.

To structure our analysis, we first modify the solution obtained by BRICKS into a solution that uses possibly non-integral number of bags of each size. In such a solution, we can use fractions of bags (such as  $\frac{4}{5}$  of a bag of size 8 as in the Figure 5.4). We arrange the modification so that the total size of bags of cost  $z$  is exactly  $m$ , except for the smallest and largest bag costs. In the main part of our proof, we bound the rounding error, i.e., the difference between the sizes of the integral and fractional solution. To complete the proof, we calculate the total size of bags in the modified fractional solution, which is easy, and show that it is well above  $n$ .

For the fractional solutions, it is better to use an alternative representation of the bags by a function  $F$  that for each  $z$  gives the number  $F(z)$  of bags of cost  $z$ . The size of  $F$  is then defined as the total size of bags. Recall that a bag of cost  $z$  has size  $\lfloor z \cdot \rho \rfloor$ . Formally:



■ **Figure 5.3** Tabular and graphical representation of the execution of BRICKS for  $n = 45$ ,  $m = 9$  and  $\rho = 1.6$ . The numbers above bags represent their sizes. The sum of bag sizes is actually  $46 > n = 45$ , to solve this, we can for example replace one bag of size 3 with a bag of size 2.



■ **Figure 5.4** Fractional solution for  $n = 45$ ,  $m = 9$  and  $\rho = 1.6$  produced by BRICKSFRACT. Notice that we always use only one bag size (cost) between consecutive multiples of  $m$ . Compare this to Figure 5.3 where bag cost 5 “overflows” the line at  $4m$  coins.

► **Definition 5.2.** A fractional solution is a mapping  $F : \mathbb{N} \rightarrow \mathbb{R}_0^+$  satisfying  $\sum_{z=1}^{\infty} F(z) = b$ . The size of fractional solution  $F$  for robustness factor  $\rho$  is defined as

$$\text{size}(F, \rho) = \sum_{z=1}^{\infty} F(z) \cdot \lfloor z \cdot \rho \rfloor.$$

We will sometimes use only  $\text{size}(F)$  if  $\rho$  is clear from the context.

We start by reformulating BRICKS so that it produces the solution directly in the alternative representation, see Algorithm BRICKSALT below. It is easy to see that BRICKS and BRICKSALT are equivalent.

► **Observation 5.3.** BRICKS and BRICKSALT use each bag cost the same number of times.

**Proof.** One step of BRICKSALT corresponds to several steps of BRICKS. BRICKS chooses the bags one by one, and it may choose the same bag cost in several consecutive iterations. BRICKSALT in each step calculates how many bags of given cost would BRICKS use. The key observation is that the expression  $\lceil (c - m(z - 1))/z \rceil$  calculates how many bags of cost  $z$  are needed to have at most  $m(z - 1)$  coins remaining. In other words, it calculates how many bags of cost  $z$  BRICKS uses before it starts using bags of cost  $z - 1$  (or runs out of bags). Hence both BRICKS and BRICKSALT use the same number of bags of cost  $z$  for each  $z$ . ◀

Algorithm BRICKSFRACT (see below) is obtained from BRICKSALT by removing the rounding in the calculation of the number of bags  $x$ .

## 8:14 Speed-Robust Scheduling Revisited

### Algorithm BRICKSALT.

---

**Input:** number of bricks  $n$ ; number of machines  $m$ ; number of bags  $b$   
**Output:** Fractional solution  $I$

$r \leftarrow b$   $\triangleright r$  is the remaining number of bags (integral)  
 $c \leftarrow n$   $\triangleright c$  is the remaining number of coins (integral)  
 $I[z] \leftarrow 0$  for  $z \in \mathbb{N}$   
**while**  $r > 0$  **and**  $c > 0$  **do**  
     $z \leftarrow \lceil \frac{c}{m} \rceil$   $\triangleright z$  is the bag cost  
     $x \leftarrow \min\left(r, \lceil \frac{c-m(z-1)}{z} \rceil\right)$   $\triangleright x$  is the (integral) number of bags of cost  $z$   
     $r \leftarrow r - x$   
     $c \leftarrow c - x \cdot z$   
     $I[z] \leftarrow x$   
**return**  $I$

---

### Algorithm BRICKSFRACT.

---

**Input:** number of bricks  $n$ ; number of machines  $m$ ; number of bags  $b$   
**Output:** Fractional solution  $F$

$r \leftarrow b$   $\triangleright r$  is the remaining number of bags (fractional)  
 $c \leftarrow n$   $\triangleright c$  is the remaining number of coins (fractional)  
 $F[z] \leftarrow 0$  for  $z \in \mathbb{N}$   
**while**  $r > 0$  **and**  $c > 0$  **do**  
     $z \leftarrow \lceil \frac{c}{m} \rceil$   
     $x \leftarrow \min\left(r, \frac{c-m(z-1)}{z}\right)$   $\triangleright x$  is the fractional amount of bags of cost  $z$   
     $r \leftarrow r - x$   
     $c \leftarrow c - x \cdot z$   
     $F[z] \leftarrow x$   
**return**  $F$

---

The following observation says that the algorithm follows our initial intuition, namely that for bags of each cost we use exactly  $m$  coins, except for the first and last bag cost used.

► **Definition 5.4.** Let  $F$  be a fractional solution, then let  $z_{\min}$  and  $z_{\max}$  denote the smallest and largest integers such that  $F(z_{\min}) > 0$  and  $F(z_{\max}) > 0$ .

► **Observation 5.5.** Let  $F$  be a result BRICKSFRACT with input  $n$  and  $m$ . Then  $F(z) = m/z$  for every  $z$  such that  $z_{\min} < z < z_{\max}$ .

**Proof.** Observe that in every step of the algorithm, except the last one, it holds that  $x = (c - m(z - 1))/z$  and thus  $c - x \cdot z = m(z - 1)$ . It follows that in all the steps except for the first and last ones  $x = (mz - m(z - 1))/z = m/z$ . ◀

Next we observe that the result of BRICKSFRACT scales, i.e., essentially it depends only on  $\lambda$ . Note also that BRICKSFRACT is well defined even for non-integral  $m$  and  $n$ .

► **Observation 5.6.** Let  $\alpha \in \mathbb{R}^+$ . Suppose BRICKSFRACT produces solution  $F$  with  $n$  and  $m$  as an input and solution  $\bar{F}$  with inputs  $\alpha n$  and  $\alpha m$ . Then  $\bar{F}(z) = \alpha F(z)$  for all  $z$ . It follows that  $\text{size}(\bar{F}, \rho) = \alpha \cdot \text{size}(F, \rho)$ .

**Proof.** We go through the execution of BRICKSFRACT step by step. Suppose that we multiply both  $m$  and  $n$  by  $\alpha$ . Then in every iteration of the loop  $r$  is multiplied by  $\alpha$ ,  $c$  is multiplied by  $\alpha$ ,  $z$  stays the same, and  $x$  is multiplied by  $\alpha$ . ◀

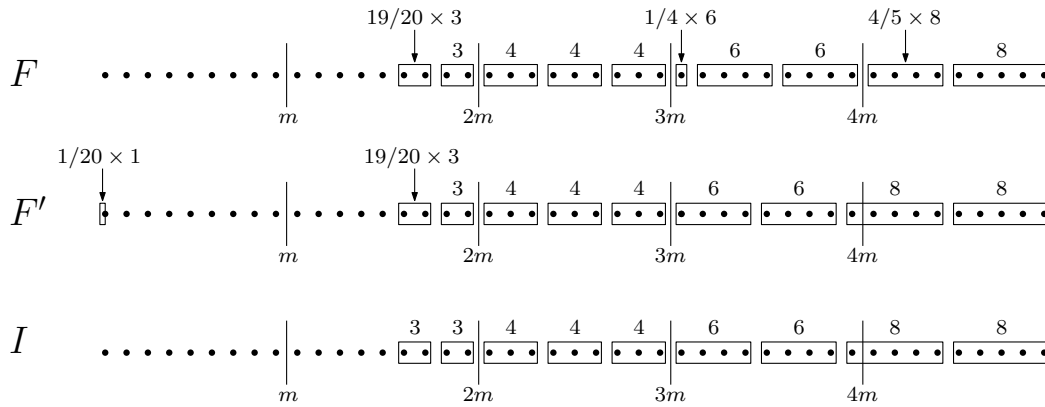
Now we are ready to bound the difference between the solutions produced by BRICKSFRACT and BRICKSALT, i.e., the rounding error.

► **Lemma 5.7.** *Let  $F$  be the fractional solution produced by BRICKSFRACT and  $I$  the solution produced by BRICKSALT on the same input. Then*

- *for  $\lambda \leq 5$  it holds that  $\text{size}(I, 1.6) \geq \text{size}(F, 1.6)$  and*
- *for  $\lambda \leq 60$  it holds that  $\text{size}(I, 1.6) \geq \text{size}(F, 1.6) - 12$ .*

**Proof.** We give an algorithm which transforms  $F$  into a solution  $F'$  that is almost integral and very close to  $I$ . Set  $\bar{z}$  to be  $z_{\min}$  of the solution  $F$  and note that  $z_{\max} = \lceil \lambda \rceil$ .

We go through the bag costs, denoted by  $z$ , from  $\lceil \lambda \rceil$  down to  $\bar{z} + 1$ . For each  $z$ , if  $F$  uses non-integral amount of bags of cost  $z$ , round it up. This makes the number of bags of cost  $z$  equal to their number in the solution  $I$ . Then decrease the number of bags of cost  $z - 1$  so that the total cost of all bags remains the same. Finally, increase the number of bags of cost 1 so that the total number of bags stays equal to  $b$ . See Figure 5.5 for an illustration.



■ **Figure 5.5** Graphical representation of  $F$ ,  $F'$  and  $I$ . In the first step of the transformation from  $F$  to  $F'$ ,  $\frac{9}{5}$  is rounded up to 2 and the number of bags of cost 5 (and size 8) increases by  $\frac{1}{5}$ . In order to keep the total cost the same, number of bags of cost 4 (and size 6) is decreased by  $\frac{1}{4}$ . As a result, total number of bags decreased by  $\frac{1}{4} - \frac{1}{5} = \frac{1}{20}$ , hence we add  $\frac{1}{20}$  of a bag of cost 1 (and size 1). This is actually the only step in which something happens since number of used bags of cost 4 and 3 is already integral. We do not process the bags of cost 2, as  $\bar{z} = 2$ . Solution  $F'$  is almost identical to the solution  $I$ , but has  $\frac{1}{20}$  of bag of cost 1 instead of  $\frac{1}{20}$  of bag of cost 2.

For  $z = \bar{z} + 1$ , the previous procedure could lead to a negative value of  $F'(\bar{z})$ . In this special case we proceed slightly differently and instead of rounding  $G(z)$  up we only increase it so that  $F'(\bar{z}) = 0$ .

We now describe one step of the process formally and analyze it. Let  $G$  denote the current fractional solution and let  $H$  denote the result of one transformation step. Let  $z$  be the current cost of bags.

We set  $H(z) = \lceil G(z) \rceil$ , note that  $H(z) - G(z) < 1$ . We want the sum of costs of bags of costs  $z - 1$  and  $z$  to remain the same, hence we want

$$H(z) \cdot z + H(z - 1) \cdot (z - 1) = G(z) \cdot z + G(z - 1) \cdot (z - 1) \tag{5.1}$$

## 8:16 Speed-Robust Scheduling Revisited

to hold. Rearranging (5.1) to an equivalent equation leads to (5.2), so we set

$$H(z-1) = G(z-1) + (G(z) - H(z)) \cdot \frac{z}{z-1}. \quad (5.2)$$

We claim that  $H(z-1) > 0$  for  $z-1 > \bar{z}$ . Indeed, as  $m > 144$  and  $z \leq \lceil \lambda \rceil \leq 60$  in the considered case, we have  $G(z-1) = F(z-1) = m/(z-1) > 2$  (using also  $\bar{z} = z_{\min} < z-1 < z_{\max}$ ). As  $H(z) - G(z) < 1$ , we get  $H(z-1) > G(z-1) - 1 > 0$ .

Now we describe the modification in the special case when  $H(z-1)$  would become negative. We have shown above that this can happen only for  $z = \bar{z} + 1$ , i.e., in the last step. Then we set  $H(z-1) = 0$  and set

$$H(z) = G(z) + (G(z-1) - H(z-1)) \frac{z-1}{z}.$$

This equation is equivalent to (5.1), which is in turn equivalent to (5.2), which thus also holds. Furthermore, the fact that the previous procedure would lead to negative  $H(z-1)$  implies that now we have  $G(z) \leq H(z) \leq \lceil G(z) \rceil$  and thus  $H(z) - G(z) < 1$  holds again.

In both cases, the total number of bags has decreased by

$$(G(z) - H(z)) + (G(z-1) - H(z-1)) = \frac{1}{z-1} (H(z) - G(z)).$$

Thus we set

$$H(1) = G(1) + \frac{1}{z-1} (H(z) - G(z)).$$

Note that in the transformation step, both the total number of bags and their total cost remain constant.

Recall that the size of a bag of cost  $z$  is  $\lfloor z\rho \rfloor$ . It follows that

$$\begin{aligned} \text{size}(H) - \text{size}(G) &= (H(z) - G(z)) \cdot \lfloor z\rho \rfloor + (H(z-1) - G(z-1)) \cdot \lfloor (z-1)\rho \rfloor + (H(1) - G(1)) \cdot \lfloor \rho \rfloor \\ &= (H(z) - G(z)) \cdot \left( \lfloor z\rho \rfloor - \frac{z}{z-1} \lfloor (z-1)\rho \rfloor + \frac{1}{z-1} \lfloor \rho \rfloor \right) \end{aligned}$$

Note that the second factor in the expression above does not depend on the solution. We call it the transformation factor and for  $z$  we denote it by

$$f(z) = \left( \lfloor z\rho \rfloor - \frac{z}{z-1} \lfloor (z-1)\rho \rfloor + \frac{1}{z-1} \lfloor \rho \rfloor \right).$$

If  $f(z) \geq 0$ , the size of the solution could have only increased, as  $H(z) \geq G(z)$ , i.e., we have  $\text{size}(H) \geq \text{size}(G)$ . If  $f(z) < 0$ , the size of the solution might have decreased – those are the important (“bad”) cases we need to bound. We have  $H(z) - G(z) < 1$ , hence  $\text{size}(H) \geq \text{size}(G) + f(z)$  in case of negative  $f(z)$ .

Now we sum these bounds over all steps for  $z$  from  $\lceil \lambda \rceil$  to 2 and get

$$\text{size}(F') - \text{size}(F) \geq \sum_{z=2}^{\lceil \lambda \rceil} \min(0, f(z))$$

We give a list of values of  $f(z)$  for  $z$  from 2 to 60 and  $\rho = 1.6$  in the full version of the paper on arXiv [8]. For  $z \leq 5$  the values  $f(z)$  are non-negative, thus for  $\lambda \leq 5$  we get  $\text{size}(F') - \text{size}(F) \geq 0$ . It can be verified that the sum of all negative values of  $f(z)$  for  $z \leq 60$  is larger than  $-12$  and thus for  $\lambda \leq 60$  we get  $\text{size}(F') - \text{size}(F) > -12$ .



Examining the algorithms BRICKSALT and BRICKSFRACT that generate the solutions  $I$  and  $F$ , respectively, and the transformation process above shows that the solution  $F$  is step by step transformed towards  $I$ . In particular,  $I(z) = F'(z)$  for all values  $z \geq \bar{z}$  (if the special case does not apply) or  $(z \geq \bar{z} + 1$  if the special case applies). For the small values of  $z$ , the only possible difference is that solution  $F'$  might have some amount of bags of size 1 instead of some larger bags in solution  $I$ . (Note that the total number of bags does not change during the transformation.) This implies  $\text{size}(I) \geq \text{size}(F')$  and the lemma follows. ◀

To complete the proof we need to show that  $\text{size}(F)$  is sufficiently large so that  $\text{size}(I) \geq n$ . Actually, as the previous transformation possibly gives  $I$  with a slightly smaller size than  $F$ , we need to compensate for this difference which is at most 12. Precisely, we need to prove that  $\text{size}(F, 1.6) \geq n$  for  $\lambda \leq 5$  and  $\text{size}(F, 1.6) \geq n + 12$  for  $5 < \lambda \leq 60$  and  $m \geq 144$ .

Since the fractional solution  $F$  scales when  $m$  and  $n$  are scaled, see Observation 5.6, it is convenient to normalize by  $m$  and consider  $(\text{size}(F, 1.6) - n)/m$  in the following lemma. Let us call this crucial quantity *normalized brick surplus*, as it measures how many bricks we are able to put in the bags in the fractional solution in addition to  $n$  bricks, normalized by  $m$ .

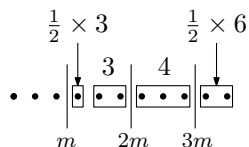
► **Lemma 5.8.** *Let  $F$  be a fractional solution produced by BRICKSFRACT. Then*

- For  $\lambda \leq 4$  it holds that  $\text{size}(F, 1.6) \geq n$ .
- For  $4 \leq \lambda \leq 60$  it holds that  $\text{size}(F, 1.6) \geq n + \frac{1}{12}m$ .

**Proof.** By Observation 5.6, the normalized brick surplus  $(\text{size}(F, 1.6) - n)/m$  is uniquely determined by  $\lambda$ , i.e., multiplying both  $n$  and  $m$  by the same constant does not change it.

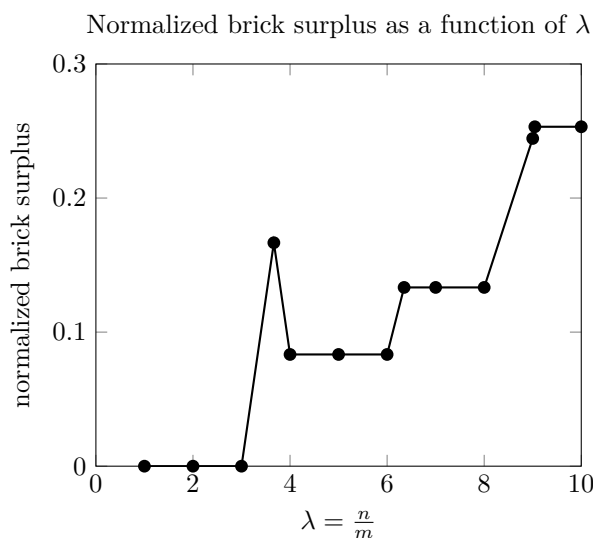
This means that the normalized brick surplus is a function of  $\lambda$ . Furthermore, we claim that the function is piece-wise linear. Suppose we slowly increase  $\lambda$ , for example fix  $m$  and increase  $n$  by  $\delta$ . Then  $F(z)$  remains constant for all  $z$  except  $z_{\min}$  and  $z_{\max}$  by Observation 5.5. The number of the largest bags  $F(z_{\max})$  increases by  $\delta/\lceil\lambda\rceil$  and  $F(z_{\min})$  decreases by the same amount; this amount is proportional to  $\delta$ . The function  $\text{size}(F)$  is linear in the values of  $F(z)$ . So the normalized brick surplus is piece-wise linear with possible breakpoints between the segments at the values of  $\lambda$  when one of the values of  $z_{\min}$  or  $z_{\max}$  changes.

The value of  $z_{\max}$  changes exactly when  $\lambda$  is an integer. The breakpoints where  $z_{\min}$  increases can be calculated in the following way: Execute BRICKSFRACT for all integer values of  $\lambda \leq 60$ . Let us denote one of such solutions  $F$ . Take a look at  $F(z_{\min})$ , if we now slowly increase  $\lambda$ ,  $F(z_{\min})$  will decrease linearly as described above. Calculate at which point it reaches 0; if it happens before  $\lambda$  increases above another integer, we found a point where  $z_{\min}$  changes. The first case of changing  $z_{\min}$  is at  $\lambda = \frac{11}{3}$  when we stop using bags of cost 1, see Figure 5.6.

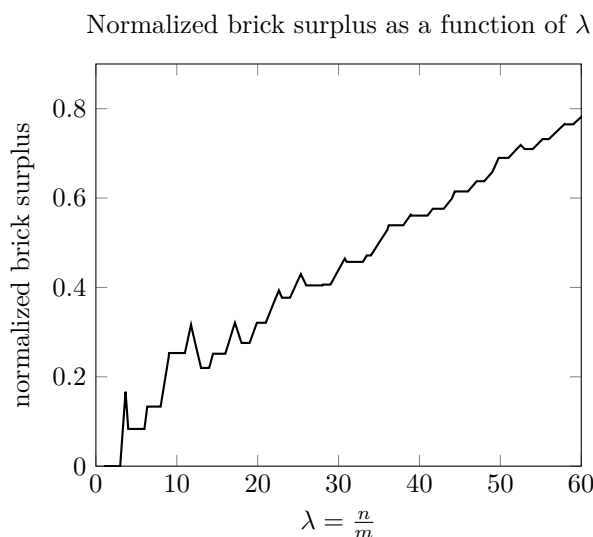


■ **Figure 5.6** Example of solution produced by BRICKSFRACT for  $n = 11$ ,  $m = 3$  and  $\rho = 1.6$ . Size of this solution is 11.5 and normalized brick surplus is  $\frac{1}{6}$ . The solution does not use any bags of cost 1. However, if  $\lambda$  were smaller, the solution would use bags of size 1.

The computer-generated tables of values of the normalized brick surplus function are given in the full version of the paper on arXiv [8]. The plot of the values is given in Figures 5.7 and 5.8 below.



■ **Figure 5.7** Plot of normalized brick surplus for small  $\lambda$ .



■ **Figure 5.8** Plot of normalized brick surplus for large  $\lambda$ .

The lemma now follows, since the normalized brick surplus is always non-negative and it is at least  $1/12$  for  $\lambda \geq 5$ . (Note that it is a constant function equal to  $1/12$  for  $\lambda \in [4, 6]$ .) ◀

► **Theorem 5.9.** *For  $\rho = 1.6$  and  $\lambda \leq 60$ , Algorithm BRICKS always succeeds, i.e., outputs bags of total size at least  $n$ .*

**Proof.** For  $\lambda \leq 5$ , the first claims in Lemmata 5.8 and 5.7 together prove  $\text{size}(I, 1.6) \geq \text{size}(F, 1.6) \geq n$ .

For  $5 \leq \lambda \leq 60$  and  $m \geq 144$  the second claims in Lemmata 5.8 and 5.7 together prove  $\text{size}(I, 1.6) \geq \text{size}(F, 1.6) - 12 \geq n + m/12 - 12 \geq n + 144/12 - 12 = n$ .

For  $\lambda \leq 60$  and  $m \leq 144$  there are only a finitely many instances and we verify  $\text{size}(I, 1.6) \geq n$  for them by computer, see the full version of the paper on arXiv [8]. ◀

We note that our choice of the bounds in the previous two lemmata is somewhat arbitrary. The plots of the normalized brick surplus suggest that we could bound it by an appropriate linear function instead of a constant. Also, the bound on  $\text{size}(F) - \text{size}(I)$  can be made smaller for intermediate values of  $\lambda$ . These changes would decrease the number of cases we need to check by a computer program, but would not improve the robustness factor.

### 5.3 Second stage

We need to show that if BRICKS succeeds, in the second stage we can indeed achieve makespan  $\rho$ . To do this, we cannot use Algorithm GREEDYASSIGNMENT and Theorem 2.1. Instead we modify it to Algorithm INTEGRALASSIGNMENT below, which copies the coins accounting scheme from BRICKS and thus follows the intuition behind it.

■ **Algorithm** INTEGRALASSIGNMENT.

---

**Input:** bag sizes  $a_1 \geq \dots \geq a_b$ ; machine speeds  $s_1 \geq \dots \geq s_m$ ; desired robustness factor  $\rho$

**for**  $i \leftarrow 1$  to  $m$  **do**

$c_i \leftarrow s_i$  ▷ Machine  $i$  gets  $s_i$  coins at the beginning.

$M_i = \emptyset$  ▷ Initialize the assignment

**for**  $k \leftarrow 1$  to  $b$  **do**

$i \leftarrow$  index of the machine with the largest  $c_i$

$M_i \leftarrow M_i \cup \{k\}$  ▷ Assign bag  $k$  to machine  $i$

$c_i \leftarrow c_i - \lceil a_k/\rho \rceil$  ▷ Machine  $i$  pays for the bag  $k$

**return**  $M_1, \dots, M_m$

---

► **Theorem 5.10.** *Suppose the first-stage algorithm BRICKS succeeds, i.e., outputs bags of total size of at least  $n$ . Then INTEGRALASSIGNMENT in the second stage produces an assignment with makespan at most  $\rho$ .*

**Proof.** Imagine that BRICKS and INTEGRALASSIGNMENT are running in parallel. BRICKS chooses the size of one bag and INTEGRALASSIGNMENT assigns it to a machine. Note that the values of  $c_i$  remain integral during the entire execution.

We claim that during the execution the value  $c$  in BRICKS is at most  $\sum_{i=1}^m c_i$  for  $c_i$ 's in INTEGRALASSIGNMENT. At the beginning, the quantities are equal. Suppose that BRICKS creates a bag of cost  $z$  and thus decreases  $c$  by  $z$ . Then the bag has size  $a = \lfloor z \cdot \rho \rfloor \leq z\rho$ . Thus INTEGRALASSIGNMENT decreases  $c_i$  by  $\lceil a/\rho \rceil \leq \lceil z\rho/\rho \rceil = z$ . Thus the sum of  $c_i$ 's decreases by at most  $z$  and the claim follows.

The claim implies that in each step before creating/assigning a bag of cost  $z$ , there exists a machine with  $c_i \geq z$  in INTEGRALASSIGNMENT. Indeed, BRICKS chooses  $z = \lceil c/m \rceil$ , thus  $m(z - 1) < c \leq \sum_{i=1}^m c_i$  using the previous claim. Hence there exists a machine with  $c_i > z - 1$  and together with integrality of  $c_i$  we get  $c_i \geq z$ .

It follows that  $c_i$ 's remain non-negative during the execution. Thus INTEGRALASSIGNMENT assigned to machine  $i$  bags of the total size at most  $s_i \cdot \rho$ . It follows that the makespan is at most  $\rho$ . ◀

Theorems 5.1, 5.9, and 5.10 immediately imply our main result.

► **Theorem 5.11.** *There exists 1.6-robust algorithm for the case of bricks and  $b = m$ .* ◀

## Conclusions

Our main result still leaves a small gap in the bounds for bricks (equal-length jobs) and  $b = m$  between the lower bound of  $e/(e - 1) \approx 1.58$  and our upper bound of 1.6. Our algorithm BRICKS does not admit a smaller robustness factor than 1.6, as is shown for  $n = 45$  and  $m = 9$  in Figure 5.3. So a smaller upper bound would need some additional techniques or special handling of some cases. Eberle *et al.* [4] give an example that shows a lower bound for bricks that is larger than  $\bar{\rho}(m, m)$  for  $m = 6$ . Although the value of the bound is below the limit value  $e/(e - 1)$ , this may be taken as a weak evidence that matching the lower bound may be hard.

The main open problem in this model remains to find a  $(2 - \varepsilon)$ -robust algorithm for the general case and  $b = m$ .

---

## References

- 1 Susanne Albers. Algorithms for dynamic speed scaling. In *Proc. of the 28th Int. Symp. on Theoretical Aspects of Computer Science, STACS 2011*, volume 9 of *LIPICs*, pages 1–11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPICs.STACS.2011.1.
- 2 Bo Chen, André van Vliet, and Gerhard J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Oper. Res. Lett.*, 18(3):127–131, 1995. doi:10.1016/0167-6377(95)00039-9.
- 3 Florian Diedrich, Klaus Jansen, Ulrich M. Schwarz, and Denis Trystram. A survey on approximation algorithms for scheduling with machine unavailability. In Jürgen Lerner, Dorothea Wagner, and Katharina Anna Zweig, editors, *Algorithmics of Large and Complex Networks – Design, Analysis, and Simulation [DFG priority program 1126]*, volume 5515 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2009. doi:10.1007/978-3-642-02094-0\_3.
- 4 Franziska Eberle, Ruben Hoeksma, Nicole Megow, Lukas Nölke, Kevin Schewior, and Bertrand Simon. Speed-robust scheduling: sand, bricks, and rocks. *Math. Program.*, 197(2):1009–1048, 2023. A preliminary version appeared at 22nd IPCO, vol 12707 of LNCS, pages 283–296, Springer, 2021. doi:10.1007/S10107-022-01829-0.
- 5 Leah Epstein, Asaf Levin, Alberto Marchetti-Spaccamela, Nicole Megow, Julián Mestre, Martin Skutella, and Leen Stougie. Universal sequencing on an unreliable machine. *SIAM J. Comput.*, 41(3):565–586, 2012. doi:10.1137/110844210.
- 6 Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988. doi:10.1137/0217033.
- 7 Csanád Imreh and John Noga. Scheduling with machine cost. In *Proc. of the 3rd Int. Workshop on Randomization and Approximation Techniques in Computer Science and 2nd Int. Workshop on Approximation Algorithms for Combinatorial Optimization Problems, RANDOM-APPROX’99*, volume 1671 of *Lecture Notes in Computer Science*, pages 168–176. Springer, 1999. doi:10.1007/978-3-540-48413-4\_18.
- 8 Josef Minařík and Jiří Sgall. Speed-robust scheduling revisited. *arXiv e-prints*, 2024. arXiv:2407.11670.
- 9 Kirk Pruhs. Speed scaling. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms – 2016 Edition*, pages 2045–2047. Springer, 2016. doi:10.1007/978-1-4939-2864-4\_390.
- 10 Kirk Pruhs, Jiri Sgall, and Eric Torng. Online scheduling. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling – Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004. doi:10.1201/9780203489802.CH15.
- 11 Clifford Stein and Mingxian Zhong. Scheduling when you do not know the number of machines. *ACM Trans. Algorithms*, 16(1):9:1–9:20, 2020. A preliminary version appeared at 29th SODA, pages 1261–1273, ACM, 2018. doi:10.1145/3340320.
- 12 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. doi:10.1017/CB09780511921735.


# On the Generalized Mean Densest Subgraph Problem: Complexity and Algorithms

Karthekeyan Chandrasekaran ✉ 


University of Illinois, Urbana-Champaign, USA

Chandra Chekuri ✉ 

University of Illinois, Urbana-Champaign, USA

Manuel R. Torres ✉ 

University of Illinois, Urbana-Champaign, USA

Weihao Zhu ✉ 

University of Illinois, Urbana-Champaign, USA

---

## Abstract

Dense subgraph discovery is an important problem in graph mining and network analysis with several applications. Two canonical polynomial-time solvable problems here are to find a *maxcore* (subgraph of maximum min degree) and to find a *densest subgraph* (subgraph of maximum average degree). Both of these problems can be solved in polynomial time. Veldt, Benson, and Kleinberg [47] introduced the generalized  $p$ -mean densest subgraph problem which captures the maxcore problem when  $p = -\infty$  and the densest subgraph problem when  $p = 1$ . They observed that for  $p \geq 1$ , the objective function is supermodular and hence the problem can be solved in polynomial time. In this work, we focus on the  $p$ -mean densest subgraph problem for  $p \in (-\infty, 1)$ . We prove that for every  $p \in (-\infty, 1)$ , the problem is NP-hard, thus resolving an open question from [47]. We also show that for every  $p \in (0, 1)$ , the weighted version of the problem is APX-hard. On the algorithmic front, we describe two simple  $\frac{1}{2}$ -approximation algorithms for every  $p \in (-\infty, 1)$ . We complement the approximation algorithms by exhibiting non-trivial instances on which the algorithms simultaneously achieve an approximation factor of at most  $\frac{1}{2}$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis

**Keywords and phrases** Densest subgraph problem, Hardness of approximation, Approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.9

**Category** APPROX

**Funding** *Karthekeyan Chandrasekaran*: partially supported by NSF grant CCF-1907937 AND CCF-2402667.

*Chandra Chekuri*: partially supported by NSF grants CCF-1907937 and CCF-2402667.

*Manuel R. Torres*: supported in part by fellowships from NSF and the Sloan Foundation, and NSF grant CCF-1910149.

*Weihao Zhu*: supported by a graduate fellowship from the CS department.

**Acknowledgements** We thank Sanjeev Khanna and Euiwoong Lee for pointers to [23] and [26] on the hardness of Exact  $\ell$ -Cover. We thank Farouk Harb for helpful discussions. This work was done when Manuel R. Torres was a student at University of Illinois, Urbana-Champaign.

## 1 Introduction

Dense subgraph discovery is an essential tool in graph mining and network analysis. The aim here is to find clusters in a graph which are denser than the entire graph. There are a number applications of dense subgraph discovery in biological settings [29, 20, 35, 4, 42], web mining [24, 14], social network analysis [34], real-time story identification [2], and finance



© Karthekeyan Chandrasekaran, Chandra Chekuri, Manuel R. Torres, and Weihao Zhu; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 9; pp. 9:1–9:21



Leibniz International Proceedings in Informatics

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

and fraud detection [15, 48, 31]. Motivated by the needs of applications and theoretical considerations, various density measures have been used and studied in the literature (see [18, 25, 37, 45, 36] for some surveys). Each density definition leads to a corresponding combinatorial optimization problem: given a graph  $G$ , find a subgraph of maximum density. Two of the most popular density measures in the literature are (i) the minimum degree of the subgraph and (ii) the average degree of the subgraph. These measures lead to the maxcore problem and densest subgraph problem respectively. They are both polynomial-time solvable and have been extensively studied. We briefly describe them before discussing a common generalization that will be the focus of this work.

In the maxcore problem (MAXCORE), the goal is to find a subgraph with maximum minimum degree. The optimum value of this problem is known as the *degeneracy* of the graph and the subgraph achieving the optimum is known as a *maxcore* of the graph. A *k-core* of a graph is a maximal connected subgraph in which all vertices have degree at least  $k$ . Min-degree is a popular measure of density, commonly finding use in what is known as the *k-core decomposition*, a nested sequence of subgraphs that captures *k*-cores for every  $k$ . One nice feature of a *k-core decomposition* is that there is a simple linear-time peeling algorithm to compute it. The peeling algorithm – denoted **Greedy** – produces an ordering of the vertices by repeatedly removing the vertex with least degree in the current graph. This ordering can in turn be used to solve MAXCORE. We refer the reader to [38] for a survey on *k-core decomposition* and applications.

In the densest subgraph problem (DSG), the goal is to find a subgraph of maximum average degree. DSG is widely used in graph mining applications. It is a well-studied problem in combinatorial optimization and is polynomial time solvable via a variety of techniques including network flow [40, 21], submodular function minimization (folklore), and linear programming [9]. Even though DSG can be solved exactly, the algorithms are slow and this has spurred the design of fast approximation algorithms [9, 5, 13, 8, 7, 10, 27]. Amongst these approximation algorithms is a peeling algorithm introduced by Asahiro, Iwama, Tamaki, and Tokuyama [3] which was shown to be a  $\frac{1}{2}$ -approximation by Charikar [9]. We note that the peeling order of this algorithm is the same as the one for computing a maxcore, namely **Greedy**; a second step of the algorithm returns the best subgraph induced by a suffix of the peeling order (best in terms of average degree). The specific density measure for DSG is used only in the second step. Charikar’s analysis has spurred the development and analysis of a variety of peeling algorithms for several variants of DSG in both graphs and hypergraphs [1, 46, 44, 30, 33, 47].

Veldt, Benson, and Kleinberg [47] introduced the generalized mean densest subgraph problem that unifies MAXCORE and DSG. The input here is a real value  $p \in \mathbb{R} \cup \{-\infty, \infty\}$  and an undirected graph  $G = (V, E)$ . For a subset  $S \subseteq V$  of vertices, the density of the subgraph  $G[S]$  induced by  $S$  is defined as:

$$M_p(S) := \left( \frac{1}{|S|} \sum_{v \in S} d_S(v)^p \right)^{1/p},$$

where  $d_S(v)$  is the degree of vertex  $v$  in the subgraph  $G[S]$ . We note that  $M_{-\infty}(S) = \min_{v \in S} d_S(v)$  is the minimum degree in the induced subgraph  $G[S]$ , while  $M_{\infty}(S) = \max_{v \in S} d_S(v)$  is the maximum degree. For  $p = 0$ , the density of the subgraph  $G[S]$  is  $M_0(S) = (\prod_{v \in S} d_S(v))^{1/|S|} = \exp(\frac{1}{|S|} \sum_{v \in S} \ln d_S(v))$ . The goal is to find a subset  $S$  of vertices with maximum  $M_p(S)$ . We refer to this problem as the  $p$ -mean densest subgraph problem ( $p$ -MEAN DSG). As  $p$  varies from  $-\infty$  to  $\infty$ ,  $M_p(S)$  prioritizes the smallest degree in  $S$  to the largest degree in  $S$  and consequently,  $p$ -MEAN DSG provides a smooth way to generate subgraphs with different density properties.

$p$ -MEAN DSG generalizes to weighted graphs in a natural manner. For a graph  $G = (V, E)$  with positive edge weights  $w : E \rightarrow \mathbb{R}_+$ , we define  $d_S(v)$  as the sum of the weight of edges that are incident to vertex  $v$  in  $G[S]$ . For a subset  $S \subseteq V$  of vertices, its  $p$ -mean density  $M_p(S)$  is defined using  $d_S(v)$  as it was for the unweighted case. The goal again is to find a subset  $S$  of vertices with maximum  $M_p(S)$ . We refer to this problem as WEIGHTED  $p$ -MEAN DSG.

Veldt, Benson, and Kleinberg made several contributions to  $p$ -MEAN DSG. They observed that 1-MEAN DSG is equivalent to DSG and that  $(-\infty)$ -MEAN DSG is equivalent MAXCORE. For  $p \geq 1$ , they observed that the set function  $f_p : 2^V \rightarrow \mathbb{R}_{\geq 0}$  defined by  $f_p(S) := \sum_{v \in S} d_S(v)^p$  is a supermodular set function<sup>1</sup>. This implies that one can solve  $p$ -MEAN DSG in polynomial time for all  $p \geq 1$  via a standard reduction to submodular set function minimization, a classical polynomial-time solvable problem in combinatorial optimization [41]. Motivated by the fact that exact algorithms are very slow in practice, they described a greedy peeling algorithm, denoted Greedy- $p$ , that runs in  $O(mn)$  time and achieves an approximation factor of  $1/(p+1)^{1/p}$  for  $p \geq 1$  (here  $m$  and  $n$  are the number of edges and nodes of the graph). The peeling order of their Greedy- $p$  algorithm is *not* the same as that of Greedy – in particular, the peeling order depends on  $p$ . They supplement these theoretical results with empirical evaluation, showing that Greedy- $p$  returns solutions with desirable characteristics for values of  $p$  in the range  $[1, 2]$ . We note that the function  $f_p(S)$  is *not* supermodular if  $p < 1$ , which partially stems from the fact that the univariate function  $g(x) := x^p$  is not convex if  $p < 1$ .

## 1.1 Our Results

We study the complexity and algorithmic status of  $p$ -MEAN DSG for  $p \in (-\infty, 1)$  which was mentioned as a compelling direction for future work by Veldt et al. [47]. It is intriguing that  $p$ -MEAN DSG is polynomial-time solvable for  $p = -\infty$  and  $p \geq 1$  while the status for  $p \in (-\infty, 1)$  is non-trivial to understand. Our work fills this gap.

### Hardness of $p$ -mean DSG for $p \in (-\infty, 1)$

We prove that  $p$ -MEAN DSG is NP-Hard for every  $p \in (-\infty, 1)$ . We also show that WEIGHTED  $p$ -MEAN DSG is APX-hard for every fixed constant  $p \in (0, 1)$ . The hardness results are the main contribution of this work. They are technically involved for two reasons. First, the objective function is non-linear and does not fall into a clean and known class of functions. Second, the problem is effectively an unconstrained problem. The initial inspiration for our reduction came from a high-level connection to submodularity due to the concavity of the univariate function  $x^p$  for  $p \in (0, 1)$ ; constrained versions of submodular optimization are NP-hard. However, the objective function for  $p$ -MEAN DSG is not a submodular function and there are no constraints. Nevertheless, we are able to model it via a gadget. Although the reduction is quite simple to describe, the proof of the reduction requires careful parameter setting and a detailed case analysis. The reduction/analysis for the weighted case is somewhat easier, however, we prove NP-Hardness for the unweighted case since it is of particular interest. We prove APX-hardness for the weighted case, and only for  $p \in (0, 1)$ , to mitigate the calculations. It may be possible to extend our APX-hardness proof to the unweighted case and also to the full range  $(-\infty, 1)$ .

<sup>1</sup> A real-valued set function  $f : 2^V \rightarrow \mathbb{R}$  is *supermodular* if  $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$  for all  $A, B \subseteq V$ . We recall that  $f$  is supermodular iff  $-f$  is *submodular*.

### Approximation algorithms for $p \in (-\infty, 1)$

The NP-Hardness result for  $p$ -MEAN DSG motivates the search for approximation algorithms for  $p \in (-\infty, 1)$ . We note that the peeling algorithm for  $p$ -MEAN DSG, namely Greedy- $p$ , given by Veldt, Benson, and Kleinberg [47] is well-defined only for  $p > 0$ . In the same paper, the authors show empirical results for Greedy- $p$  for  $p \in (0, 1)$  even though the corresponding function  $f_p$  is not supermodular; however, no approximation guarantee is known for Greedy- $p$  for  $p \in (0, 1)$ . We describe two different and simple algorithms for  $p$ -MEAN DSG— one based on simple greedy peeling and the other based on an exact solution to DSG. We show that both algorithms achieve a  $\frac{1}{2}$ -approximation for all  $p \in (-\infty, 1)$ . These are the first algorithms with approximation guarantees for  $p$  in the regime  $(-\infty, 1)$ . We complement the algorithms by exhibiting tight instances on which *both* algorithms exhibit an approximation factor of at most  $\frac{1}{2}$ , thus ruling out the possibility of improving the ratio by taking the best of the two algorithms.

This paper builds upon and extends an earlier version [11] by two of the authors. The previous version included results on faster algorithms for  $p$ -MEAN DSG for  $p > 1$  and empirical evaluation of several algorithms. A full version including those results will be made available in the future.

### Organization

We present the NP-hardness result in Section 2 and the APX-hardness result in Section 3. We present our algorithmic results in Section 4. All proofs that are omitted from the main body of the paper are given in the appendix.

### Notation

Let  $G = (V, E)$  be a graph. For a subset  $S \subseteq V$  of vertices and a vertex  $v \in S$ , we recall that  $d_S(v)$  is the (weighted) degree of  $v$  in the induced subgraph  $G[S]$ . Let  $S$  be a subset of vertices. We define  $f_p(S) := \sum_{v \in S} d_S(v)^p$  if  $p \in [-\infty, 0) \cup (0, \infty]$  and  $f_p(S) := \sum_{v \in S} \ln d_S(v)$  if  $p = 0$ . We also define  $\rho_p(S) := f_p(S)/|S|$  for all  $p$ . With these definitions, we have  $M_p(S) = \rho_p(S)^{1/p}$  for all  $p$ . Thus, finding a set  $S$  of vertices with maximum  $M_p(S)$  is equivalent to finding a set  $S$  of vertices with *maximum*  $\rho_p(S)$  if  $p \geq 0$ , and to finding a set  $S$  of vertices with *minimum*  $\rho_p(S)$  if  $p < 0$ .

## 1.2 Other Related Work

DSG and the subfield of dense subgraph discovery is large. We point the reader to a recent survey [36] and restrict our attention to discussing some closely related work, specifically on sequential models and approximability.

As we remarked, DSG is poly-time solvable by several techniques including via maximum flow. Although maximum flow now admits an almost-linear time algorithm [12], the existing exact algorithms for DSG are slow in practice for large graphs. Thus approximation algorithms have also been considered (especially before the recent developments on network flow). In particular, there has been a line of work that obtained a  $(1 - \epsilon)$ -approximation in  $\tilde{O}(m \cdot \text{poly}(\frac{1}{\epsilon}))$ -time [5, 8, 10]; in particular, the algorithm in [10] runs in time  $\tilde{O}(\frac{m}{\epsilon})$ . These faster approximation algorithms also have some limitations in practice for large graphs. Several simpler iterative algorithms based on continuous optimization methods have been developed – these include algorithms based on the Frank-Wolfe method [13], an algorithm based on iterating Greedy called Greedy++ [7], and the projected gradient descent method [27].



They are simple to implement and have been shown to converge quickly to near-optimal solutions on large graphs, both synthetic and real-world, even though the known worst-case theoretical convergence rates are fairly large.

We discuss some other measures of density considered in the literature. Given a graph  $G = (V, E)$  and a finite collection of pattern graphs  $\mathcal{F}$ , one can consider the problem of finding a set  $S$  that maximizes  $f(S)/|S|$  where  $f : 2^V \rightarrow \mathbb{Z}_+$  counts the number of occurrences of the patterns from  $\mathcal{F}$  in the induced subgraph  $G[S]$ . If we consider a single edge as the pattern, then we obtain DSG. [46] considered the special case where  $\mathcal{F}$  is a single triangle graph and [44] considered the special case where  $\mathcal{F}$  is a clique on  $k$  vertices. The densest subgraph problem under the general notion of patterns was considered in [17]. Densest subgraph has also been studied for hypergraphs with density of a set  $S$  of vertices being defined as  $|E(S)|/|S|$  where  $E(S)$  is the set of hyperedges with all vertices in  $S$  [30]. We can reduce the densest subgraph problem with pattern based densities to the densest subgraph problem in hypergraphs by introducing a hyperedge for each occurrence of the pattern in the input graph. Charikar's analysis of Greedy can be generalized to show an approximation factor of at least  $\frac{1}{r}$  in rank  $r$ -hypergraphs. Veldt, Benson, and Kleinberg [47] showed that Greedy is not a good worst-case algorithm for  $p$ -MEAN DSG when  $p > 1$ , and as we mentioned earlier, they developed a different peeling algorithm. Chekuri, Quanrud and Torres [10] unified several results by considering density measures of the form  $f(S)/|S|$  where  $f : 2^V \rightarrow \mathbb{R}_+$  is an arbitrary non-negative supermodular set function over a vertex set  $V$ . They showed that there is a natural peeling algorithm for each  $f$  and derived an approximation bound in terms of certain properties of  $f$ ; Greedy- $p$  from [47] and its approximation bound are derived as special cases. They also generalized Greedy++ to supermodular densities and showed that the resulting algorithm converges to an optimum solution, partially answering a conjecture from [7] (the conjecture has a strong convergence rate). See [27, 28] for additional insights. One can also consider density measures of the form  $f(S)/g(S)$  where  $g : 2^V \rightarrow \mathbb{R}_+$  is another set function such as a concave or convex function of  $|S|$ . We refer the reader to [36, 10] for results and pointers on this aspect.

Constrained versions of DSG such as the  $k$ -densest-subgraph (find a densest subgraph with at most  $k$  vertices) are well-studied in theoretical computer science.  $k$ -densest-subgraph is of particular importance due to its connection to various other problems, and due to the intriguing difficulty in understanding its approximability. Since there is a large literature on this problem and since constrained versions are not the focus of this paper, we point the reader to some relevant papers on algorithms and hardness [19, 6, 39].

## 2 NP-hardness

In this section, we prove the following theorem.

► **Theorem 1.**  *$p$ -MEAN DSG is NP-hard for all  $p \in (-\infty, 1)$ .*

We reduce from the Exact  $\ell$ -Cover problem.

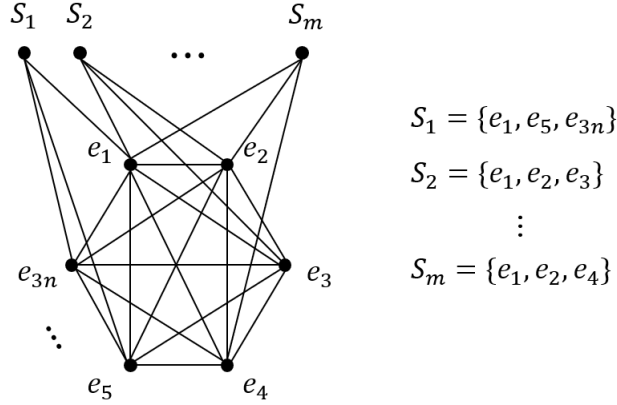
► **Problem 1.** *EXACT  $\ell$ -COVER: the input is a finite ground set  $\mathcal{U} = \{e_1, e_2, \dots, e_{\ell n}\}$  of cardinality  $\ell n$  for some positive integers  $\ell$  and  $n$ , and a family of subsets  $\mathcal{S} \subseteq 2^{\mathcal{U}}$ , where each  $X \in \mathcal{S}$  has cardinality  $\ell$ . The goal is to determine whether there exist  $n$  disjoint sets  $S_{i_1}, S_{i_2}, \dots, S_{i_n} \in \mathcal{S}$  whose union is  $\mathcal{U}$ .*

We will say that the input instance  $(\mathcal{U}, \mathcal{S})$  of EXACT  $\ell$ -COVER has an exact  $\ell$ -cover if there exist  $n$  disjoint sets whose union is  $\mathcal{U}$ . EXACT 3-COVER is a well-known NP-complete problem [22]. A standard padding approach reduces EXACT 3-COVER to EXACT  $\ell$ -COVER for every  $\ell \geq 3$ .

► **Theorem 2.** *EXACT  $\ell$ -COVER is NP-complete for every integer  $\ell \geq 3$ .*

### 2.1 Reduction from exact $\ell$ -cover

We reduce EXACT  $\ell$ -COVER to  $p$ -MEAN DSG. Let  $\mathcal{U} = \{e_1, e_2, \dots, e_{\ell n}\}$  and  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  with  $S_i \subseteq \mathcal{U}$  for every  $i \in [m]$  be the input instance of EXACT  $\ell$ -COVER. For a positive integer  $d$  (to be chosen later), we construct a graph  $G_d = (L \cup A, E)$ , where  $L := \{v_i : i \in [m]\}$  and  $A := \{u_j : j \in [\ell \cdot n]\}$ . For every  $1 \leq i \leq m$  and  $1 \leq j \leq \ell \cdot n$ , if the set  $S_i$  contains element  $e_j$ , then we add an edge between  $v_i$  and  $u_j$  in graph  $G_d$ . Further, we add edges between vertices in  $A$  to make  $G_d[A]$  to be a connected  $d$ -regular graph, where  $d$  will be chosen appropriately (we may assume that  $n$  is even so that such a connected  $d$ -regular graph always exists). See Figure 1. If  $p \neq 0$ , then we set  $\rho^* := \frac{\ell^p + \ell \cdot (d+1)^p}{\ell+1}$  and if  $p = 0$ , then we set  $\rho^* := \frac{\ln \ell + \ell \cdot \ln(d+1)}{\ell+1}$ . We will show that there exist positive integers  $\ell \geq 3$  and  $d$  such that the input instance admits an exact  $\ell$ -cover if and only if  $\max\{M_p(X) : X \subseteq V\} \geq (\rho^*)^{1/p}$ .



■ **Figure 1** Graph constructed in our reduction from EXACT  $\ell$ -COVER for  $\ell = 3$  and  $d = 5$ . The EXACT 3-COVER instance consists of the ground set  $\mathcal{U} := \{e_1, \dots, e_{3n}\}$  and the family  $\mathcal{S} := \{S_1, \dots, S_m\}$ .

Next, we prove the NP-hardness of  $p$ -MEAN DSG by casing on the value of  $p$  via the above mentioned reduction. We prove NP-hardness for  $p \in (0, 1)$  in Section 2.2 (see Theorem 7). The missing proofs are given in the appendix. The proofs for  $p \in (-\infty, 0]$  are given in the full version owing to space limitations.

### 2.2 NP-hardness for $p \in (0, 1)$

We recall that  $\rho_p(X) = M_p(X)^p$  for every subset  $X$  of vertices and hence, finding a set  $X$  of vertices that maximizes  $M_p(X)$  reduces to finding a set  $X$  of vertices that maximizes  $\rho_p(X)$  if  $p \in (0, 1)$ . We define  $\text{OPT}_{G_d} := \max_{X \subseteq V} \rho_p(X)$ . We observe that  $\text{OPT}_{G_d}$  is a maximization problem for  $p \geq 0$ . Hence, in order to show correctness of our reduction, we will need an upper bound on  $\rho_p(X)$  for  $p \geq 0$ . The following lemma gives an upper bound.

► **Lemma 3.** *Let  $G_d = (L \cup A, E)$  be the graph constructed in the reduction. Let  $S \subseteq L$  and  $A' \subseteq A$ . Then, for  $p > 0$ , we have that*

$$\rho_p(S \cup A') \leq \frac{\ell^p \cdot |S| + \sum_{v \in A'} (d + d_{S+v}(v))^p}{|S| + |A'|}.$$

Moreover, the inequality above is strict if  $|A'| < |A|$ .

**Proof.** For the case of  $p > 0$ , we note that

$$\begin{aligned}
\rho_p(S \cup A') &= \frac{f_p(S \cup A')}{|S \cup A'|} \\
&= \frac{\sum_{u \in S} d_{S \cup A'}(u)^p + \sum_{v \in A'} d_{S \cup A'}(v)^p}{|S| + |A'|} \\
&\leq \frac{\sum_{u \in S} d_{S \cup A}(u)^p + \sum_{v \in A'} d_{S \cup A}(v)^p}{|S| + |A'|} \quad (\text{since } p > 0) \\
&= \frac{\ell^p \cdot |S| + \sum_{v \in A'} d_{S \cup A}(v)^p}{|S| + |A'|} \quad (\text{since } d_{S \cup A}(u) = \ell) \\
&= \frac{\ell^p \cdot |S| + \sum_{v \in A'} (d + d_{S+v}(v))^p}{|S| + |A'|}. \quad (\text{since } G[A] \text{ is a } d\text{-regular graph})
\end{aligned}$$

If  $|A'| < |A|$ , then there exists a vertex  $u \in A'$  such that  $d_{A'}(u) < d_A(u) = d$  because  $G[A]$  is connected, which implies that the inequality above is strict.  $\blacktriangleleft$

We need the following technical lemma about the maximizer of a relevant function.

► **Lemma 4.** *Let  $c \in \mathbb{R}_{\geq 0}$ . Let  $f : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}$  be defined as  $f(x) := \sum_{i=1}^n (c + x_i)^p$ . For  $0 < p < 1$ , consider the following maximization problem parameterized by  $s \in \mathbb{N}$ :*

$$\text{maximize } \left\{ f(x) : x \in \mathbb{Z}^n, x \geq 0, \sum_{i=1}^n x_i \leq s \right\}.$$

*Every maximizer for the above problem has  $\mu$  coordinates set to  $\lceil s/n \rceil - 1$  and  $n - \mu$  coordinates set to  $\lceil s/n \rceil$ , where  $\mu = n \cdot \lceil s/n \rceil - s$ . If  $s$  is a multiple of  $n$ , then the maximizer has all coordinates set to  $s/n$ .*

**Proof.** Let  $x \in \mathbb{Z}^n$  and  $x \geq 0$  be a maximizer. If  $\sum_{i=1}^n x_i < s$ , then by setting  $x'_n = x_n + 1$  and  $x'_k = x_k$  for every  $k \in [n - 1]$ , we have  $f(x') > f(x)$ , a contradiction to optimality of  $f(x)$ .

Suppose that  $\sum_{i=1}^n x_i = s$ . We prove that all coordinates are in  $\{\lceil s/n \rceil, \lceil s/n \rceil - 1\}$ . Assume that  $x$  has at least one entry not in  $\{\lceil s/n \rceil, \lceil s/n \rceil - 1\}$ . Then, there exists some coordinate  $x_i$  that is strictly larger than  $\lceil s/n \rceil$  or smaller than  $\lceil s/n \rceil - 1$ . Without loss of generality, we assume that  $x_i > \lceil s/n \rceil$ . Consequently, there exists some index  $j \in [n]$  such that  $x_j < \lceil s/n \rceil$ . Since  $x_j < \lceil s/n \rceil < x_i$  and  $x_i, x_j$  are both integers, we have  $x_i - x_j \geq 2$ . Because  $(c + x)^p$  is a concave function of  $x$ ,  $(c + x_i)^p + (c + x_j)^p < (c + x_i - 1)^p + (c + x_j + 1)^p$ . By setting  $x'_i = x_i - 1$ ,  $x'_j = x_j + 1$ , and  $x'_k = x_k$  for every  $k \in [n] \setminus \{i, j\}$ , we have  $f(x') > f(x)$ , a contradiction to optimality of  $f(x)$ .

Hence, for every maximizer  $x \in \mathbb{Z}^n$ , we have  $\sum_{i=1}^n x_i = s$  and  $x_i \in \{\lceil s/n \rceil, \lceil s/n \rceil - 1\}$  for every  $i \in [n]$ . This implies that  $(n - \mu)$  coordinates are  $\lceil s/n \rceil$  and  $\mu$  coordinates are  $\lceil s/n \rceil - 1$ , where  $\mu = n \cdot \lceil s/n \rceil - s$ . When  $s$  is a multiple of  $n$ , by Jensen's inequality, we have  $x_i = s/n$  for every  $1 \leq i \leq n$ .  $\blacktriangleleft$

We will use the following lemma about the existence of an integer  $\ell$  that satisfies two inequalities simultaneously for every given  $p$ .

► **Lemma 5.** *For every  $p \in (0, 1)$ , there exists an integer  $\ell \geq 3$  s.t the following two inequalities hold:*

$$\left(1 - \frac{1}{2\ell}\right)^p < 1 - \frac{1 - 1/2^p}{\ell + 1} \quad \text{and} \quad \left(1 + \frac{1}{2\ell}\right)^p < 1 + \frac{1 - 1/2^p}{\ell + 1}. \quad (1)$$

We need the following lemma about NO-instances of EXACT  $\ell$ -COVER.

► **Lemma 6.** *Let  $p \in (0, 1)$  and  $\ell \geq 3$  be an integer that satisfies the two inequalities in (1). Consider an instance of EXACT  $\ell$ -COVER with ground set  $\mathcal{U}$  of size  $\ell n$  and family  $\mathcal{S} \subseteq 2^{\mathcal{U}}$ . Suppose that the instance has no exact  $\ell$ -cover. Then, for every non-negative integers  $s, a' \in \mathbb{Z}_{\geq 0}$  with  $s \leq |\mathcal{S}|$ ,  $a' \leq \ell n$ , and  $s + a' \geq 1$  and every non-negative integer vector  $X \in \mathbb{Z}_{\geq 0}^{a'}$  with  $\sum_{i=1}^{a'} X_i \leq \ell s$ , we have that*

$$\frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} \leq \frac{\ell^p + \ell \cdot (2\ell)^p}{\ell + 1}. \quad (2)$$

Moreover, if there exists  $i \in [a']$  such that  $X_i \neq \ell s/a'$ , then the above inequality is strict.

**Proof.** We case on the value of the ratio  $s/a'$ .

**Case 1.** Suppose  $\ell \cdot s = a'$ . Then, we have

$$\begin{aligned} \frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} &\leq \frac{\ell^p \cdot s + a' \cdot (2\ell - 1 + \frac{\ell \cdot s}{a'})^p}{s + a'} \quad (\text{by } \sum_{i=1}^{a'} X_i \leq \ell \cdot s \text{ and Lemma 4}) \\ &= \frac{\ell^p \cdot a'/\ell + a' \cdot (2\ell)^p}{a'/\ell + a'} \quad (\text{since } \ell \cdot s = a') \\ &= \frac{\ell^p + \ell \cdot (2\ell)^p}{\ell + 1}. \end{aligned}$$

By Lemma 4, if there exists  $i \in [a']$  such that  $X_i \neq \ell s/a'$ , then the above inequality is strict.

**Case 2.** Suppose  $\ell \cdot s = \beta \cdot a'$  for some  $0 \leq \beta < 1$ . Then, we have

$$\begin{aligned} \frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} &\leq \frac{\ell^p \cdot s + \beta a' \cdot (2\ell)^p + (1 - \beta)a' \cdot (2\ell - 1)^p}{s + a'} \\ &\quad (\text{by } \sum_{i=1}^{a'} X_i \leq \ell \cdot s \text{ and Lemma 4}) \\ &= \frac{\ell^p \cdot \beta + \ell \beta \cdot (2\ell)^p + \ell(1 - \beta) \cdot (2\ell - 1)^p}{\beta + \ell}. \quad (\text{since } \ell \cdot s = \beta \cdot a') \end{aligned}$$

We define  $h : [0, 1] \rightarrow \mathbb{R}$  as

$$h(\beta) := \frac{\ell^p \cdot \beta + \ell \beta \cdot (2\ell)^p + \ell(1 - \beta) \cdot (2\ell - 1)^p}{\beta + \ell},$$

which implies that the left hand side expression in the lemma is at most  $h(\beta)$ . By differentiating the function  $h$  with respect to  $\beta$ , we have

$$\begin{aligned} \frac{d}{d\beta} h(\beta) &= \frac{(\ell^p + \ell \cdot (2\ell)^p - \ell \cdot (2\ell - 1)^p)(\beta + \ell) - (\ell^p \cdot \beta + \ell \beta \cdot (2\ell)^p + \ell(1 - \beta) \cdot (2\ell - 1)^p)}{(\beta + \ell)^2} \\ &= \frac{\ell^{p+1} + \ell^2 \cdot (2\ell)^p - \ell^2 \cdot (2\ell - 1)^p - \ell \cdot (2\ell - 1)^p}{(\beta + \ell)^2} \\ &= \frac{\ell^{p+1} + \ell^2 \cdot (2\ell)^p - (\ell^2 + \ell) \cdot (2\ell - 1)^p}{(\beta + \ell)^2} \\ &> \frac{\ell^{p+1} + \ell^2 \cdot (2\ell)^p - (\ell^2 + \ell) \cdot (2\ell)^p \cdot (1 - \frac{1-2^{-p}}{\ell+1})}{(\beta + \ell)^2} \quad (\text{by inequality (1)}) \\ &= \frac{\ell^{p+1} + \ell^2 \cdot (2\ell)^p - \ell \cdot (2\ell)^p \cdot (\ell + 2^{-p})}{(\beta + \ell)^2} = 0. \end{aligned}$$

Hence, function  $h(\beta)$  is strictly increasing for  $\beta \in [0, 1]$ . Thus,

$$\frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} \leq h(\beta) < h(1) = \frac{\ell^p + \ell \cdot (2\ell)^p}{\ell + 1}.$$

**Case 3.** Suppose  $\ell \cdot s = \alpha \cdot a'$  for some  $\alpha > 1$ . Let  $\alpha = t + \beta$  where  $t \geq 1$  is an integer and  $0 < \beta \leq 1$ . Then, we have

$$\begin{aligned} \frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} &\leq \frac{\ell^p \cdot s + \beta a' \cdot (2\ell + t)^p + (1 - \beta) |A'| \cdot (2\ell - 1 + t)^p}{|S| + |A'|} \\ &\quad (\text{by } \sum_{i=1}^{a'} X_i \leq \ell \cdot s \text{ and Lemma 4}) \\ &= \frac{\ell^p \cdot (t + \beta) + \ell \beta \cdot (2\ell + t)^p + \ell(1 - \beta) \cdot (2\ell - 1 + t)^p}{t + \beta + \ell} \\ &\quad (\text{since } \ell \cdot s = (t + \beta) \cdot a'). \end{aligned}$$

We define  $g : [0, +\infty) \times [0, 1] \rightarrow \mathbb{R}$  as

$$g(t, \beta) := \frac{\ell^p \cdot (t + \beta) + \ell \beta \cdot (2\ell + t)^p + \ell(1 - \beta) \cdot (2\ell - 1 + t)^p}{t + \beta + \ell},$$

which implies that the left hand side expression in the lemma is at most  $g(t, \beta)$ . We note that  $g(t, 1) = g(t + 1, 0)$  for every  $t \geq 0$ .

By differentiating the function  $g$  with respect to  $\beta$ , we have

$$\begin{aligned} \frac{d}{d\beta} g(t, \beta) &= \frac{(\ell^p + \ell \cdot (2\ell + t)^p - \ell \cdot (2\ell - 1 + t)^p)(t + \beta + \ell) - (\ell^p \cdot (t + \beta) + \ell \beta \cdot (2\ell + t)^p + \ell(1 - \beta) \cdot (2\ell - 1 + t)^p)}{(t + \beta + \ell)^2} \\ &= \frac{\ell^{p+1} + (\ell^2 + t \cdot \ell) \cdot (2\ell + t)^p - (\ell^2 + (t + 1) \cdot \ell) \cdot (2\ell + t - 1)^p}{(t + \beta + \ell)^2}. \end{aligned}$$

Now, we note that

$$\begin{aligned} \frac{d}{dt} (\ell^{p+1} + (\ell^2 + t \cdot \ell) \cdot (2\ell + t)^p - (\ell^2 + (t + 1) \cdot \ell) \cdot (2\ell + t - 1)^p) &= p \cdot (\ell^2 + t \cdot \ell) \cdot (2\ell + t)^{p-1} + \ell \cdot (2\ell + t)^p - p \cdot (\ell^2 + (t + 1) \cdot \ell) \cdot (2\ell + t - 1)^{p-1} - \ell \cdot (2\ell + t - 1)^p \\ &\leq p \cdot (\ell^2 + t \cdot \ell) \cdot (2\ell + t)^{p-1} + \ell \cdot p \cdot (2\ell + t - 1)^{p-1} - p \cdot (\ell^2 + (t + 1) \cdot \ell) \cdot (2\ell + t - 1)^{p-1} \\ &\quad (\text{since } (x + 1)^p - x^p \leq p \cdot x^{p-1} \text{ for every } x > 0) \\ &= p \cdot (\ell^2 + t \cdot \ell) \cdot ((2\ell + t)^{p-1} - (2\ell + t - 1)^{p-1}) < 0. \end{aligned}$$

Thus,

$$\begin{aligned} \frac{d}{d\beta} g(t, \beta) &= \frac{\ell^{p+1} + (\ell^2 + t \cdot \ell) \cdot (2\ell + t)^p - (\ell^2 + (t + 1) \cdot \ell) \cdot (2\ell + t - 1)^p}{(t + \beta + \ell)^2} \\ &\leq \frac{\ell^{p+1} + (\ell^2 + \ell) \cdot (2\ell + 1)^p - (\ell^2 + 2\ell) \cdot (2\ell)^p}{(1 + \beta + \ell)^2} \quad (\text{since } t \geq 1) \\ &= \frac{(2\ell)^p \cdot (2^{-p} \cdot \ell + (\ell^2 + \ell) \cdot (1 + \frac{1}{2\ell})^p) - (\ell^2 + 2\ell)}{(1 + \beta + \ell)^2} \\ &< \frac{(2\ell)^p \cdot (2^{-p} \cdot \ell + (\ell^2 + \ell) \cdot (1 + \frac{1-2^{-p}}{\ell+1})) - (\ell^2 + 2\ell)}{(1 + \beta + \ell)^2} \quad (\text{by inequality (1)}) \\ &= \frac{(2\ell)^p \cdot (2^{-p} \cdot \ell + (\ell^2 + \ell) + (1 - 2^{-p}) \cdot \ell) - (\ell^2 + 2\ell)}{(1 + \beta + \ell)^2} = 0. \end{aligned}$$

## 9:10 On the Generalized Mean Densest Subgraph Problem: Complexity and Algorithms

Hence, function  $g(t, \beta)$  is strictly decreasing with respect to  $\beta$  for  $\beta \in [0, 1]$ . Thus,

$$\frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} \leq g(t, \beta) < g(t, 0).$$

For every positive integer  $r > 1$ , we have  $g(r, 0) = g(r - 1, 1) < g(r - 1, 0)$ . Thus,

$$\frac{\ell^p \cdot s + \sum_{i=1}^{a'} (2\ell - 1 + X_i)^p}{s + a'} < g(t, 0) < g(1, 0) = \frac{\ell^p + \ell \cdot (2\ell)^p}{\ell + 1}. \quad \blacktriangleleft$$

Now, we are ready to prove the NP-hardness of  $p$ -MEAN DSG for  $p \in (0, 1)$ . We recall that  $\text{OPT}_G := \max_{X \subseteq V} \rho_p(X)$ .

► **Theorem 7.** *For every  $p \in (0, 1)$ , there exist positive integers  $\ell \geq 3$  and  $d$  such that for an instance of EXACT  $\ell$ -COVER with ground set  $\mathcal{U}$  and family  $\mathcal{S} \subseteq 2^{\mathcal{U}}$ , there exists an exact  $\ell$ -cover iff  $\text{OPT}_G \geq \rho^* = \frac{\ell^p + \ell \cdot (d+1)^p}{\ell + 1}$ , where  $G = G_d$  is the graph constructed in the reduction above.*

**Proof.** By Corollary 5, we know that there exists a positive integer  $\ell \geq 3$  satisfying the two inequalities in (1). Fix such an  $\ell$ . Let  $\mathcal{U}$  be the ground set and  $\mathcal{S}$  be the collection of subsets of an instance of EXACT  $\ell$ -COVER. We set  $d = 2\ell - 1$  and consider the graph  $G = G_d$  constructed in the reduction in Section 2.1. We show that there exists an exact  $\ell$ -cover iff  $\text{OPT}_G \geq \rho^* = \frac{\ell^p + \ell \cdot (d+1)^p}{\ell + 1}$ .

Suppose  $\mathcal{S}$  contains an exact  $\ell$ -cover  $S_{i_1}, \dots, S_{i_n}$ . Let  $S = \{v_{i_1}, v_{i_2}, \dots, v_{i_n}\}$ . We note that  $|S| = n = \frac{|A|}{\ell}$ . Thus, we have

$$\text{OPT}_G \geq \rho_p(S \cup A) = \frac{f_p(S \cup A)}{|S \cup A|} = \frac{\ell^p \cdot |S| + (d+1)^p \cdot |A|}{|S| + |A|} = \frac{\ell^p \cdot |A|/\ell + (d+1)^p \cdot |A|}{|A|/\ell + |A|} = \rho^*.$$

Suppose  $\mathcal{S}$  does not contain an exact  $\ell$ -cover. Let  $S \subseteq L$  and  $A' \subseteq A$ . We note that  $\ell \cdot |S| = \sum_{v \in A} d_{S+v}(v) \geq \sum_{v \in A'} d_{S+v}(v)$ . Thus, we have

$$\begin{aligned} \rho_p(S \cup A') &= \frac{f_p(S \cup A')}{|S \cup A'|} \leq \frac{\ell^p \cdot |S| + \sum_{v \in A'} (d + d_{S+v}(v))^p}{|S| + |A'|} \quad (\text{by Lemma 3}) \\ &\leq \rho^*. \quad (\text{by Lemma 6}) \end{aligned}$$

We note that since  $S$  is not an exact  $\ell$ -cover, we have that either  $|A'| < |A|$  or there exists  $u, v \in A$  with  $d_{S+u}(u) \neq d_{S+v}(v)$ . This implies that either the first inequality or the second inequality above is strict according to the respective lemmas, that is,  $\rho_p(S \cup A') < \rho^*$ . ◀

### 3 APX-hardness for $p \in (0, 1)$

In this section, we adapt the NP-hardness proof from Section 2 to show that WEIGHTED  $p$ -MEAN DSG is APX-hard for every fixed constant  $p \in (0, 1)$ .

► **Theorem 8.** *For every fixed constant  $p \in (0, 1)$ , there exists a constant  $\delta_p > 0$  that depends only on  $p$  such that it is NP-hard to obtain a  $(1 - \delta_p)$ -approximation for WEIGHTED  $p$ -MEAN DSG.*

In order to prove Theorem 8, we will rely on the APX-hardness of EXACT  $\ell$ -COVER as stated below.

► **Theorem 9.** *There exists a constant  $\varepsilon \in (0, 1)$  such that for every integer  $\ell \geq 3$ , it is NP-hard to distinguish between the following two cases for a given finite ground set  $\mathcal{U}$  of size  $\ell n$  and a family  $\mathcal{S} \subseteq 2^{\mathcal{U}}$  of subsets each of cardinality  $\ell$ :*

- *YES-instance: There exists a collection of  $n$  sets in  $\mathcal{S}$  whose union is  $\mathcal{U}$ .*
- *NO-instance: The union of every collection of  $n \cdot (1 + \varepsilon)$  sets in  $\mathcal{S}$  has size at most  $\ell \cdot n \cdot (1 - \varepsilon)$ .*

Exact  $\ell$ -Cover is a special case of Set Cover. The hardness we seek requires a disjoint set cover in the YES case, and we also need the hardness to hold for every fixed integer  $\ell \geq 3$ . Related results have been proved in the literature [32, 43, 26, 23], however the precise version we need requires a formal argument. We provide the proof in the full version and also comment on the relation to previous work.

### Reduction from Exact $\ell$ -Cover to weighted $p$ -mean DSG

We reduce from the APX-hard variant of EXACT  $\ell$ -COVER, namely the problem mentioned in Theorem 9. Consider an instance of the problem mentioned in Theorem 9: namely, let  $\mathcal{U}$  be a ground set of size  $\ell n$  and let  $\mathcal{S} \subseteq 2^{\mathcal{U}}$  of subsets each of cardinality  $\ell$ . For a positive integer  $d$  (to be chosen later), we construct a graph  $G_d = (L \cup A, E)$  as follows: we define  $L := \{v_i : i \in [m]\}$  and  $A := \{u_j : j \in [\ell \cdot n]\}$ . For every  $i \in [m]$  and  $j \in [\ell \cdot n]$ , if set  $S_i$  contains element  $e_j$ , then we add an edge with unit weight between  $v_i$  and  $u_j$  in graph  $G$ . We add an edge between all pairs of vertices in  $A$  with weight  $\frac{d}{|A|-1}$ , where  $d$  will be chosen appropriately (instead of  $G[A]$  being a connected  $d$ -regular graph as used in the NP-hardness reduction in Section 2). We note that for every vertex  $v \in A$ , the sum of weight of edges incident to  $v$  in the induced subgraph  $G[A]$  is  $d$ . We define  $\text{OPT}_{G_d} := \max_{X \subseteq V} \rho_p(X)$  and set  $\rho^* := \frac{\ell^p + \ell \cdot (d+1)^p}{\ell+1}$ . We will prove that if the instance is a YES instance, then  $\text{OPT}_{G_d} \geq \rho^*$  and if the instance is a NO instance, then  $\text{OPT}_{G_d} < (1 - \delta_p) \cdot \rho^*$  for some constant  $\delta_p > 0$  that depends only on  $p$ . We now state the main theorem of the section below.

► **Theorem 10.** *For every  $p \in (0, 1)$ , there exist positive integers  $\ell \geq 3$  and  $d$  such that for an instance  $(\mathcal{U}, \mathcal{S})$  of the problem mentioned in Theorem 9, where the ground set  $\mathcal{U}$  has size  $\ell n$  and every set in  $\mathcal{S}$  has size  $\ell$ , the following two hold:*

- *if the instance is a YES-instance, then  $\text{OPT}_{G_d} \geq \rho^*$ , and*
- *if the instance is a NO-instance, then  $\text{OPT}_{G_d} < (1 - \delta_p) \cdot \rho^*$  for some constant  $\delta_p > 0$  that depends only on  $p$ .*

Here,  $G_d$  is the graph constructed in the reduction from Exact  $\ell$ -Cover for Weighted Version.

Theorem 8 follows from Theorem 10. We briefly outline our proof of Theorem 10 and refer to the full version for the full proof. It is easy to see that if the instance  $(\mathcal{U}, \mathcal{S})$  of the problem mentioned in Theorem 9 is a YES-instance, then  $\text{OPT}_{G_d} \geq \rho^*$  (similar to the proof of NP-hardness). We focus on showing that if the instance is a NO-instance, then  $\text{OPT}_{G_d} < (1 - \delta) \rho^*$ . Let  $S \subseteq L$  and  $A' \subseteq A$ . We need to show that  $\rho_p(S \cup A') < (1 - \delta) \rho^*$ . For this, we recall the proof of NP-hardness in Section 2.2. There, we showed that if the instance does not have an exact  $\ell$ -cover, then  $\rho_p(S \cup A') < \rho^*$ . For this, we proved that  $\rho_p(S \cup A')$  is maximized and is at most  $\rho^*$  if  $\ell|S|/|A'| = 1$ . That proof can be adapted in a straightforward fashion to show that  $\rho_p(S \cup A') < (1 - \delta) \rho^*$  if  $\ell|S|/|A'| \geq 1 + \varepsilon$  or if  $\ell|S|/|A'| \leq 1 - \varepsilon$  for some constants  $\delta, \varepsilon > 0$  (even for the graph  $G_d$  that appears in the reduction to unweighted  $p$ -MEAN DSG) – see cases 1 and 2 in the proof of Theorem 10. Thus, the non-trivial case to handle is if  $\ell|S|/|A'| \in (1 - \varepsilon, 1 + \varepsilon)$ . In this situation, we consider two cases: (i) Suppose that  $|A'| \leq (1 - \varepsilon)|A|$ . In this case, we exploit the clique in the weighted graph constructed in the reduction above to conclude that  $\rho_p(S \cup A') < (1 - \delta) \rho^*$  for some constant  $\delta > 0$  (see case 3 in the proof of Theorem 10). (ii) Suppose that  $|A'| > (1 - \varepsilon)|A|$ .

In this case, we rely on the APX-hardness of EXACT  $\ell$ -COVER (i.e., the instance  $(\mathcal{U}, \mathcal{S})$  is a NO-instance of the problem mentioned in Theorem 9) to conclude that  $\rho_p(S \cup A') < (1 - \delta)\rho^*$  for some constant  $\delta > 0$  (see case 4 in the proof of Theorem 10). We emphasize that the weighted clique over the set  $A$  of vertices in the reduction graph (as opposed to an unweighted  $d$ -regular graph over the set  $A$  of vertices) is useful in the first case. We also mention that the constant  $\delta_p$  in Theorem 10 is very small. We give an estimation of  $\delta_{1/2}$  in the full version.

## 4 Approximation Algorithms

We give two new approximation algorithms for  $p$ -MEAN DSG. Our algorithms achieve an approximation factor of  $\frac{1}{2}$  for all  $p \in (-\infty, 1)$ . Our algorithms rely on the fact that MAXCORE and DSG can be solved in polynomial time. First, we show that the peeling algorithm used to compute MAXCORE can be adapted to obtain a  $\frac{1}{2}$ -approximate solutions to  $p$ -MEAN DSG for every  $p \in (-\infty, 1)$ . Secondly, we show that an optimum solution to DSG is a  $\frac{1}{2}$ -approximate solution to  $p$ -MEAN DSG for every  $p \in (-\infty, 1)$ . We complement these results with a family of graphs for which both algorithms *simultaneously* achieve only a  $\frac{1}{2}$ -approximation.

Let  $G = (V, E)$  be the input graph. We let  $S_p^* := \arg \max_{S \subseteq V} M_p(S)$  and let  $M_p^* := M_p(S_p^*)$ . We need the following fact about the monotonicity of the objective.

► **Proposition 11.** *Let  $S \subseteq V$ . For every  $p \leq q$ , we have  $M_p(S) \leq M_q(S)$ .*

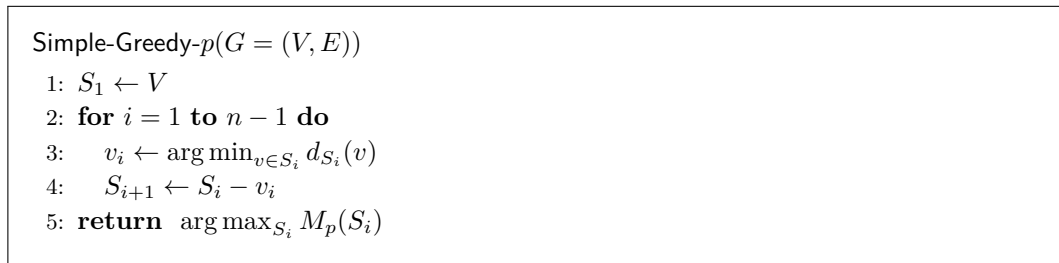
We have the following statement connecting different values of  $M_p^*$ .

► **Proposition 12.** *For every  $p \in [-\infty, 1]$ , we have  $M_{-\infty}^* \leq M_p^* \leq M_1^* \leq 2M_{-\infty}^*$ .*

The first two inequalities follow directly from Proposition 11 and the last inequality follows via a simple known argument connecting degeneracy to the maximum average degree of a subgraph (e.g., see [16]).

### $\frac{1}{2}$ -approximation via maxcore approach

Our first algorithm leverages the standard greedy peeling algorithm for the maxcore. Our algorithm, denoted Simple-Greedy- $p$ , is given in Figure 2. The algorithm for  $p = -\infty$  is the peeling algorithm used to compute maxcore and the algorithm for  $p = 1$  is Charikar's greedy peeling algorithm. We recall that Charikar showed that the algorithm achieves a  $\frac{1}{2}$ -approximation for 1-MEAN DSG.



■ **Figure 2**  $\frac{1}{2}$ -approximation via greedy peeling for  $p$ -MEAN DSG where  $p < 1$ .



► **Theorem 13.** *Let  $p \in [-\infty, 1]$  and let  $S$  be the output of Simple-Greedy- $p(G)$ . Then,  $M_p(S) \geq \frac{1}{2}M_p^*$ .*

**Proof.** The algorithm for  $p = -\infty$  is the peeling algorithm used to compute maxcore. In particular, it is well-known that there exists  $i \in [n]$  with  $M_{-\infty}(S_i) = M_{-\infty}^*$ . By Proposition 11,  $M_{-\infty}(S_i) \leq M_p(S_i)$  and by choice of  $S$ , we have  $M_p(S_i) \leq M_p(S)$ . Therefore,  $M_{-\infty}^* \leq M_p(S)$ . Finally, by Proposition 12, we have  $\frac{1}{2}M_p^* \leq M_{-\infty}^*$ . Combining these two statements, we get  $\frac{1}{2}M_p^* \leq M_p(S)$ . ◀

► **Remark 14.** Simple-Greedy- $p$  returns an optimum solution for  $p = \infty$ . Our results show that for  $p \in (-\infty, 1]$ , Simple-Greedy- $p$  returns a  $\frac{1}{2}$ -approximation. However, for  $p > 1$ , [47] showed that the approximation factor of Simple-Greedy- $p$  can be arbitrarily small.

### $\frac{1}{2}$ -approximation via 1-mean densest subgraph

Our second algorithm is to simply return a 1-mean densest subgraph. We recall that  $S_1^* = \arg \max_{S \subseteq V} M_1(S)$  and it can be computed in polynomial time. We analyze its approximation factor.

► **Theorem 15.** *Let  $p \in [-\infty, 1]$ . Then,  $M_p(S_1^*) \geq \frac{1}{2}M_p^*$ .*

**Proof.** We first prove that

$$M_{-\infty}(S_1^*) \geq \frac{1}{2}M_1^*. \quad (3)$$

It suffices to show that  $d_{S_1^*}(v) \geq \frac{|E(S_1^*)|}{|S_1^*|}$  for every  $v \in S_1^*$ . Suppose towards a contradiction that there exists  $v \in S_1^*$  such that  $d_{S_1^*}(v) < \frac{|E(S_1^*)|}{|S_1^*|}$ . Using this and observing  $|E(S_1^*)| - |E(S_1^* - v)| = d_{S_1^*}(v)$ , after rearranging, we have  $\frac{|E(S_1^* - v)|}{|S_1^* - v|} > \frac{|E(S_1^*)|}{|S_1^*|}$ . Multiplying through by 2, we obtain  $M_1(S_1^* - v) > M_1(S_1^*)$ , contradicting the optimality of  $S_1^*$ .

Thus, we have

$$M_p(S_1^*) \geq M_{-\infty}(S_1^*) \geq \frac{1}{2}M_1^* \geq \frac{1}{2}M_p^*$$

where the first and last inequality are by Proposition 11 and the second inequality is via (3). ◀

► **Remark 16.** We described two algorithms that achieve an approximation factor of  $\frac{1}{2}$ . Would returning the best among the sets returned by the two algorithms achieve a factor that is better than  $\frac{1}{2}$ ? In the full version, we construct a non-trivial family of instances on which both algorithms have an approximation factor of at most  $\frac{1}{2}$ . We emphasize that we seek non-trivial instances – in particular, instances in which the optimum value is arbitrary (i.e., grows) and is not a fixed constant.

## 5 Conclusion

MAXCORE and DSG are polynomial-time solvable densest subgraph problems with numerous applications.  $p$ -MEAN DSG, introduced by Veldt, Benson, and Kleinberg [47], captures both these special cases and provides a unified way to generate subgraphs with different density properties.  $p$ -MEAN DSG is polynomial-time solvable for  $p = -\infty$  and for  $p \geq 1$ . In this work, we addressed the complexity and algorithmic aspects of the problem for  $p \in (-\infty, 1)$ .

We showed that  $p$ -MEAN DSG is NP-hard for  $p \in (-\infty, 1)$  and WEIGHTED  $p$ -MEAN DSG is APX-hard for every fixed constant  $p \in (0, 1)$ . Our hardness results motivate the need for approximation algorithms for  $p \in (-\infty, 1)$ . We gave a simple  $1/2$ -approximation for  $p$ -MEAN DSG for all  $p \in (-\infty, 1)$ . Our approximation algorithms also extend to WEIGHTED  $p$ -MEAN DSG with the same approximation guarantee in a natural manner.

There are two interesting directions for future work. Firstly, is  $p$ -MEAN DSG (or WEIGHTED  $p$ -MEAN DSG) APX-hard for every  $p \in (-\infty, 1)$ ? Our APX-hardness results hold for every fixed constant  $p \in (0, 1)$ . Extending our approach to show APX-hardness for fixed constant  $p \in (-\infty, 0)$  requires extending the proof of Theorem 10 to  $p < 0$ . The technical barrier to extending is the third case in the proof. Secondly, can we improve the approximability of  $p$ -MEAN DSG for  $p \in (-\infty, 1)$ ? In contrast to the densest subgraph problem, the non-linearity of the objective function of  $p$ -MEAN DSG makes it difficult to develop mathematical programming relaxations. We leave it here as an interesting open problem.

---

## References

- 1 Reid Andersen and Kumar Chellapilla. Finding dense subgraphs with size bounds. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 25–37. Springer, 2009.
- 2 Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB journal*, 23:175–199, 2014.
- 3 Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.
- 4 Gary D Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4(1):1–27, 2003.
- 5 Bahman Bahmani, Ashish Goel, and Kamesh Munagala. Efficient primal-dual graph algorithms for MapReduce. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 59–78. Springer, 2014.
- 6 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.
- 7 Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*, pages 573–583, 2020.
- 8 Digvijay Boob, Saurabh Sawlani, and Di Wang. Faster width-dependent algorithm for mixed packing and covering LPs. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 2019.
- 9 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.
- 10 Chandra Chekuri, Kent Quanrud, and Manuel R Torres. Densest subgraph: Supermodularity, iterative peeling, and flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1555. SIAM, 2022.
- 11 Chandra Chekuri and Manuel R. Torres. On the generalized mean densest subgraph problem: Complexity and algorithms. *CoRR*, abs/2306.02172, 2023. doi:10.48550/arXiv.2306.02172.
- 12 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022.

- 13 Maximilien Danisch, T.-H. Hubert Chan, and Mauro Sozio. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web*, pages 233–242, 2017.
- 14 Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense communities in the web. In *Proceedings of the 16th international conference on World Wide Web*, pages 461–470, 2007.
- 15 Xiaoxi Du, Ruoming Jin, Liang Ding, Victor E Lee, and John H Thornton Jr. Migration motif: a spatial-temporal pattern mining approach for financial markets. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2009.
- 16 Martin Farach-Colton and Meng-Tsung Tsai. Computing the degeneracy of large graphs. In *LATIN 2014: Theoretical Informatics: 11th Latin American Symposium, Montevideo, Uruguay, March 31–April 4, 2014. Proceedings 11*, pages 250–260. Springer, 2014.
- 17 András Faragó. A general tractable density concept for graphs. *Mathematics in Computer Science*, 1(4):689–699, 2008.
- 18 András Faragó and Zohre R Mojaveri. In search of the densest subgraph. *Algorithms*, 12(8):157, 2019.
- 19 Uriel Feige, David Peleg, and Guy Kortsarz. The dense  $k$ -subgraph problem. *Algorithmica*, 29(3):410–421, 2001.
- 20 Eugene Fratkin, Brian T Naughton, Douglas L Brutlag, and Serafim Batzoglou. MotifCut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14):e150–e157, 2006.
- 21 Giorgio Gallo, Michael D Grigoriadis, and Robert E Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.
- 22 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.
- 23 Naveen Garg, Sanjeev Khanna, and Amit Kumar. Hardness of approximation for orienteering with multiple time windows. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2977–2990. SIAM, 2021.
- 24 David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*, pages 721–732, 2005.
- 25 Aristides Gionis and Charalampos E Tsourakakis. Dense subgraph discovery: KDD 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2313–2314, 2015.
- 26 Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michał Włodarczyk. Losing treewidth by separating subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1749. SIAM, 2019.
- 27 Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. Faster and scalable algorithms for densest subgraph and decomposition. In *Advances in Neural Information Processing Systems*, 2022.
- 28 Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. Convergence to lexicographically optimal base in a (contra)polymatroid and applications to densest subgraph and tree packing. In *31st Annual European Symposium on Algorithms*, volume 274, pages 56:1–56:17, 2023.
- 29 Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21(suppl\_1):i213–i221, 2005.
- 30 Shuguang Hu, Xiaowei Wu, and TH Hubert Chan. Maintaining densest subsets efficiently in evolving hypergraphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 929–938, 2017.

- 31 Yingsheng Ji, Zheng Zhang, Xinlei Tang, Jiachen Shen, Xi Zhang, and Guangwen Yang. Detecting cash-out users via dense subgraphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 687–697, 2022.
- 32 Viggo Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.
- 33 Yasushi Kawase and Atsushi Miyauchi. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12):3461–3480, 2018.
- 34 Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 611–617, 2006.
- 35 Tommaso Lanciano, Francesco Bonchi, and Aristides Gionis. Explainable classification of brain networks via contrast subgraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3308–3318, 2020.
- 36 Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzino, and Francesco Bonchi. A survey on the densest subgraph problem and its variants. *arXiv preprint arXiv:2303.14467*, 2023.
- 37 Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, 2010.
- 38 Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, 29:61–92, 2020.
- 39 Pasin Manurangsi. Almost-polynomial ratio ETH-hardness of approximating densest  $k$ -subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–961, 2017.
- 40 Jean-Claude Picard and Maurice Queyranne. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks*, 12(2):141–159, 1982.
- 41 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
- 42 Victor Spirin and Leonid A Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the national Academy of sciences*, 100(21):12123–12128, 2003.
- 43 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 453–461, 2001.
- 44 Charalampos Tsourakakis. The  $k$ -clique densest subgraph problem. In *Proceedings of the 24th international conference on world wide web*, pages 1122–1132, 2015.
- 45 Charalampos Tsourakakis and Tianyi Chen. Dense subgraph discovery: Theory and application (Tutorial at SDM 2021), 2021. URL: <https://tsourakakis.com/dense-subgraph-discovery-theory-and-applications-tutorial-sdm-2021/>.
- 46 Charalampos E Tsourakakis. A novel approach to finding near-cliques: The triangle-densest subgraph problem. *arXiv preprint arXiv:1405.1477*, 2014.
- 47 Nate Veldt, Austin R Benson, and Jon Kleinberg. The generalized mean densest subgraph problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1604–1614, 2021.
- 48 Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 570–578. SIAM, 2017.

## A Missing proofs for NP-hardness

We include the missing proofs for  $p \in (0, 1)$  from Section 2 here.

### A.1 Proof of Theorem 2

► **Theorem 2.** *EXACT  $\ell$ -COVER is NP-complete for every integer  $\ell \geq 3$ .*

**Proof.** We recall that EXACT 3-COVER is NP-complete [22]. We reduce EXACT  $\ell$ -COVER to EXACT  $(\ell+1)$ -COVER for  $\ell \geq 3$ . Consider an instance of EXACT  $\ell$ -COVER with ground set  $\mathcal{U}$  of cardinality  $\ell n$  and a family  $\mathcal{S} \subseteq 2^{\mathcal{U}}$  of subsets each of which has cardinality  $\ell$ . Let  $x_1, \dots, x_n$  be  $n$  new elements that are not in  $\mathcal{U}$ . We create an instance of EXACT  $(\ell+1)$ -COVER as follows:

- Let  $\mathcal{U}' := \mathcal{U} \cup \{x_1, \dots, x_n\}$  be the ground set. We have that  $|\mathcal{U}'| = |\mathcal{U}| + n = (\ell+1) \cdot n$ .
- Let  $\mathcal{S}' := \{S \cup \{x_i\} : S \in \mathcal{S}, 1 \leq i \leq n\}$ . Each set in  $\mathcal{S}'$  has cardinality  $\ell+1$ .

If there exists an exact  $\ell$ -cover  $\{S_{i_1}, S_{i_2}, \dots, S_{i_n}\}$  of  $\mathcal{U}$ , then  $\{S_{i_1} \cup \{x_1\}, S_{i_2} \cup \{x_2\}, \dots, S_{i_n} \cup \{x_n\}\}$  is an exact  $(\ell+1)$ -cover of  $\mathcal{U}$ . If an exact  $\ell$ -cover of  $\mathcal{U}$  does not exist, then an exact  $(\ell+1)$ -cover of  $\mathcal{U}'$  does not exist. Hence, NP-completeness of EXACT 3-COVER implies NP-completeness of EXACT  $\ell$ -COVER for every  $\ell \geq 3$ . ◀

### A.2 Technical Lemmas for Hardness Results

The following inequalities will be used when proving the hardness results.

► **Lemma 17.** *Let  $p, x \in (0, 1)$ . Then,*

1.  $(1-x)^p < 1 - px$ ,
2.  $(1-x)^p < 1 - px - \frac{p(1-p)}{2}x^2$ ,
3.  $(1+x)^p < 1 + px$ , and
4.  $(1+x)^p < 1 + px - \frac{p(1-p)}{2}x^2 + \frac{p(1-p)(2-p)}{6}x^3$ .

**Proof.** For the first inequality, let  $f_1(x) := (1-x)^p - (1-px)$ . Then,  $f_1'(x) = p \cdot (1 - (1-x)^{p-1}) < 0$ , which implies that  $f_1(x) < f_1(0) = 0$ .

For the second inequality, let  $f_2(x) := (1-x)^p - (1-px - \frac{p(1-p)}{2}x^2)$ . Then,  $f_2'(x) = p \cdot (1 + (1-p)x - (1-x)^{p-1})$  and  $f_2''(x) = p \cdot (1-p) \cdot (1 - (1-x)^{p-2}) < 0$ . Hence,  $f_2'(x) < f_2'(0) = 0$ , which implies that  $f_2(x) < f_2(0) = 0$ .

For the third inequality, let  $f_3(x) := (1+x)^p - (1+px)$ . Then,  $f_3'(x) = p \cdot ((1+x)^{p-1} - 1) < 0$ , which implies that  $f_3(x) < f_3(0) = 0$ .

For the fourth inequality, let  $f_4(x) := (1+x)^p - (1+px - \frac{p(1-p)}{2}x^2 + \frac{p(1-p)(2-p)}{6}x^3)$ . Then,  $f_4'(x) = p \cdot ((1+x)^{p-1} - (1 + (1-p)x + \frac{(1-p)(2-p)}{2}x^2))$ . Also,  $f_4''(x) = p \cdot (1-p) \cdot (1 - (2-p)x - (1+x)^{p-2})$  and  $f_4'''(x) = p(1-p)(2-p) \cdot ((1+x)^{p-3} - 1) < 0$ , which implies that  $f_4''(x) < f_4''(0) = 0$ . Thus,  $f_4'(x) < f_4'(0) = 0$  and  $f_4(x) < f_4(0) = 0$ . ◀

### A.3 Proof of Lemma 5

Lemma 5 follows from the following stronger lemma. The stronger version will be useful in proving APX-hardness.

► **Lemma 18.** *For every  $p \in (0, 1)$ , there exists a positive value  $\eta > 0$  and an integer  $\ell \geq 3$ , both of which depend only on  $p$ , such that the following two inequalities hold:*

$$\left(1 - \frac{1}{2\ell}\right)^p < 1 - \frac{1 - 1/2^p}{\ell + 1} - \eta \quad \text{and} \quad \left(1 + \frac{1}{2\ell}\right)^p < 1 + \frac{1 - 1/2^p}{\ell + 1} - \eta.$$

**9:18 On the Generalized Mean Densest Subgraph Problem: Complexity and Algorithms**

**Proof.** Set  $\ell_0 := \frac{p \cdot 2^{p-1}}{2^p - p \cdot 2^{p-1} - 1}$ . Then, we have  $\frac{p}{2\ell_0} = \frac{1-1/2^p}{\ell_0+1}$ . We note that

$$\ell_0 - \frac{5}{2} = \frac{p \cdot 2^{p-1}}{2^p - p \cdot 2^{p-1} - 1} - \frac{5}{2} = \frac{2^{1-p} + \frac{7}{5}p - 2}{2^p - p \cdot 2^{p-1} - 1} \cdot \frac{5}{2} \cdot 2^{p-1} = \frac{2^{1-p} - \frac{7}{5}(1-p) - \frac{3}{5}}{2^p - p \cdot 2^{p-1} - 1} \cdot \frac{5}{2} \cdot 2^{p-1} > 0.$$

The last inequality holds since  $2^x - \frac{7}{5}x - \frac{3}{5}$  is a convex function and evaluates to zero at  $x = 1$  with negative derivative. This implies that  $2^{1-p} - \frac{7}{5}(1-p) - \frac{3}{5} > 0$ . Consequently,

$$\ell_0 > \frac{5}{2}. \tag{4}$$

We also note that

$$\begin{aligned} 2 - (1-p)\ell_0 &= 2 - (1-p) \cdot \frac{p \cdot 2^{p-1}}{2^p - p \cdot 2^{p-1} - 1} \quad (\text{since } \ell_0 = \frac{p \cdot 2^{p-1}}{2^p - p \cdot 2^{p-1} - 1}) \\ &= \frac{2^p \cdot (2 - \frac{3}{2}p + \frac{1}{2}p^2 - 2^{1-p})}{2^p - p \cdot 2^{p-1} - 1} \\ &= \frac{2^p \cdot (1 + (1-p) + \frac{1}{2}(-p)(1-p) - 2^{1-p})}{2^p - p \cdot 2^{p-1} - 1} < 0. \end{aligned}$$

The last inequality above is because  $2^x > 1 + x + \frac{1}{2}x(x-1)$  for all  $x \in (0, 1)$ . This implies that

$$(1-p)\ell_0 > 2. \tag{5}$$

Hence, if we can prove that the two inequalities of the lemma hold for every  $\ell \in [\ell_0 - \frac{1}{2}, \ell_0 + \frac{1}{2}]$ , then it implies the lemma. We define two functions  $f_1 : (0, +\infty) \rightarrow \mathbb{R}$  and  $f_2 : (0, +\infty) \rightarrow \mathbb{R}$  as

$$f_1(\ell) = (1 - \frac{1}{2\ell})^p + \frac{1-1/2^p}{\ell+1} - 1 \quad \text{and} \quad f_2(\ell) = (1 + \frac{1}{2\ell})^p - \frac{1-1/2^p}{\ell+1} - 1.$$

By setting

$$\eta := \frac{1}{2} \cdot \min \left\{ \frac{\frac{1}{2}p \cdot (2\ell_0 + 3)}{\ell_0 \cdot (2\ell_0 + 3) \cdot (2\ell_0 + 1)^2} \cdot ((1-p)\ell_0 - 2), \frac{p}{2\ell_0 - 1} \cdot \frac{16\ell_0 - 16}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2} \right\},$$

which is larger than 0 by inequalities (4) and (5), we will prove that  $f_1(\ell) < -\eta$  and  $f_2(\ell) < -\eta$  for every  $\ell \in [\ell_0 - \frac{1}{2}, \ell_0 + \frac{1}{2}]$ . We note that

$$\eta < \frac{\frac{1}{2}p \cdot (2\ell_0 + 3)}{\ell_0 \cdot (2\ell_0 + 3) \cdot (2\ell_0 + 1)^2} \cdot ((1-p)\ell_0 - 2) \quad \text{and} \tag{6}$$

$$\eta < \frac{p}{2\ell_0 - 1} \cdot \frac{16\ell_0 - 16}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2}. \tag{7}$$

By differentiating  $f_1$  with respect to  $\ell$ , we have

$$\begin{aligned}
\frac{d}{d\ell} f_1(\ell) &= \frac{p}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p}} - \frac{1 - 1/2^p}{(\ell + 1)^2} \\
&= \frac{p \cdot (\ell + 1)^2 - 2(1 - 1/2^p)\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p}}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \\
&= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( \frac{p}{2(1 - 1/2^p)} \cdot (1 + \frac{1}{\ell})^2 - (1 - \frac{1}{2\ell})^{1-p} \right) \\
&= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( \frac{\ell_0}{\ell_0 + 1} \cdot (1 + \frac{1}{\ell})^2 - (1 - \frac{1}{2\ell})^{1-p} \right) \\
&\quad (\text{since } \frac{p}{2\ell_0} = \frac{1 - 1/2^p}{\ell_0 + 1}) \\
&> \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( \frac{\ell_0}{\ell_0 + 1} \cdot (1 + \frac{1}{\ell})^2 - (1 - \frac{1-p}{2\ell}) \right) \\
&\quad (\text{since } (1 - \frac{1}{2\ell})^{1-p} < 1 - \frac{1-p}{2\ell} \text{ according to Lemma 17}) \\
&= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell_0 \cdot (\ell + 1)^2 - (\ell_0 + 1) \cdot \ell^2 + \frac{1-p}{2}(\ell_0 + 1) \cdot \ell}{(\ell_0 + 1) \cdot \ell^2} \\
&> \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell_0 \cdot (\ell + 1)^2 - (\ell_0 + 1) \cdot \ell^2}{(\ell_0 + 1) \cdot \ell^2} \\
&= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell_0(2\ell + 1) - \ell^2}{(\ell_0 + 1) \cdot \ell^2} \\
&\geq \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{(\ell - \frac{1}{2})(2\ell + 1) - \ell^2}{(\ell_0 + 1) \cdot \ell^2} \quad (\text{since } \ell_0 \geq \ell - \frac{1}{2}) \\
&= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 - \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell^2 - \frac{1}{2}}{(\ell_0 + 1) \cdot \ell^2} \\
&> 0. \quad (\text{since } \ell \geq \ell_0 - \frac{1}{2} > 2 \text{ according to inequality (4)})
\end{aligned}$$

Hence, the function  $f_1(\ell)$  is strictly increasing for  $\ell \in [\ell_0 - \frac{1}{2}, \ell_0 + \frac{1}{2}]$ . This implies that

$$\begin{aligned}
f_1(\ell) &\leq f_1(\ell_0 + \frac{1}{2}) = (1 - \frac{1}{2\ell_0 + 1})^p + \frac{1 - 1/2^p}{\ell_0 + \frac{3}{2}} - 1 \\
&= \frac{\frac{\ell_0 + 1}{2\ell_0} \cdot p}{\ell_0 + \frac{3}{2}} + (1 - \frac{1}{2\ell_0 + 1})^p - 1 \quad (\text{since } \frac{p}{2\ell_0} = \frac{1 - 1/2^p}{\ell_0 + 1}) \\
&< \frac{p \cdot (\ell_0 + 1)}{\ell_0 \cdot (2\ell_0 + 3)} - \frac{p}{2\ell_0 + 1} - \frac{p(1-p)}{2} \cdot \frac{1}{(2\ell_0 + 1)^2} \\
&\quad (\text{since } (1-x)^p < 1 - px - \frac{p(1-p)}{2}x^2 \text{ according to Lemma 17}) \\
&= p \cdot \frac{(\ell_0 + 1)(2\ell_0 + 1)^2 - \ell_0(2\ell_0 + 3)(2\ell_0 + 1) - \frac{1-p}{2}\ell_0(2\ell_0 + 3)}{\ell_0 \cdot (2\ell_0 + 3) \cdot (2\ell_0 + 1)^2} \\
&= p \cdot \frac{(2\ell_0 + 1) - \frac{1-p}{2}\ell_0(2\ell_0 + 3)}{\ell_0 \cdot (2\ell_0 + 3) \cdot (2\ell_0 + 1)^2} \\
&< \frac{\frac{1}{2}p \cdot (2\ell_0 + 3)}{\ell_0 \cdot (2\ell_0 + 3) \cdot (2\ell_0 + 1)^2} \cdot (2 - (1-p)\ell_0) \\
&< -\eta. \quad (\text{by inequality (6)})
\end{aligned}$$

By differentiating  $f_2$  with respect to  $\ell$ , we have

$$\begin{aligned}
 \frac{d}{d\ell} f_2(\ell) &= -\frac{p}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p}} + \frac{1 - 1/2^p}{(\ell + 1)^2} \\
 &= \frac{-p \cdot (\ell + 1)^2 + 2(1 - 1/2^p)\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p}}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \\
 &= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( -\frac{p}{2(1 - 1/2^p)} \cdot (1 + \frac{1}{\ell})^2 + (1 + \frac{1}{2\ell})^{1-p} \right) \\
 &= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( -\frac{\ell_0}{\ell_0 + 1} \cdot (1 + \frac{1}{\ell})^2 + (1 + \frac{1}{2\ell})^{1-p} \right) \\
 &\quad \left( \text{since } \frac{p}{2\ell_0} = \frac{1 - 1/2^p}{\ell_0 + 1} \right) \\
 &< \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \left( -\frac{\ell_0}{\ell_0 + 1} \cdot (1 + \frac{1}{\ell})^2 + (1 + \frac{1-p}{2\ell}) \right) \\
 &\quad \left( \text{since } (1 + \frac{1}{2\ell})^{1-p} < 1 + \frac{1-p}{2\ell} \text{ according to Lemma 17} \right) \\
 &= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{(\ell_0 + 1) \cdot \ell^2 + \frac{1-p}{2}(\ell_0 + 1) \cdot \ell - \ell_0 \cdot (\ell + 1)^2}{(\ell_0 + 1) \cdot \ell^2} \\
 &< \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{(\ell_0 + 1) \cdot \ell^2 + \frac{1}{2}(\ell_0 + 1) \cdot \ell - \ell_0 \cdot (\ell + 1)^2}{(\ell_0 + 1) \cdot \ell^2} \\
 &= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell^2 - \frac{3}{2}\ell_0\ell + \frac{1}{2}\ell - \ell_0}{(\ell_0 + 1) \cdot \ell^2} \\
 &\leq \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{\ell(\ell_0 + \frac{1}{2}) - \frac{3}{2}\ell_0\ell + \frac{1}{2}\ell - \ell_0}{(\ell_0 + 1) \cdot \ell^2} \quad \left( \text{since } \ell \leq \ell_0 + \frac{1}{2} \right) \\
 &= \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{-\frac{1}{2}\ell_0\ell + \ell - \ell_0}{(\ell_0 + 1) \cdot \ell^2} \\
 &\leq \frac{2(1 - 1/2^p) \cdot \ell^2}{2\ell^2 \cdot (1 + \frac{1}{2\ell})^{1-p} \cdot (\ell + 1)^2} \cdot \frac{-\frac{1}{2}\ell_0\ell + \frac{1}{2}}{(\ell_0 + 1) \cdot \ell^2} \quad \left( \text{since } \ell \leq \ell_0 + \frac{1}{2} \right) \\
 &< 0. \quad \left( \text{since } \ell_0 > \frac{5}{2} \text{ and } \ell \geq \ell_0 - \frac{1}{2} > 2 \text{ according to inequality (4)} \right)
 \end{aligned}$$

Hence, function  $f_2(\ell)$  is strictly decreasing for  $\ell \in [\ell_0 - \frac{1}{2}, \ell_0 + \frac{1}{2}]$ . This implies that

$$\begin{aligned}
 f_2(\ell) &\leq f_2(\ell_0 - \frac{1}{2}) = \left(1 + \frac{1}{2\ell_0 - 1}\right)^p - \frac{1 - 1/2^p}{\ell_0 + \frac{1}{2}} - 1 \\
 &= -\frac{\frac{\ell_0+1}{2\ell_0} \cdot p}{\ell_0 + \frac{1}{2}} + \left(1 + \frac{1}{2\ell_0 - 1}\right)^p - 1 \quad \left( \text{since } \frac{p}{2\ell_0} = \frac{1 - 1/2^p}{\ell_0 + 1} \right) \\
 &< -\frac{\frac{\ell_0+1}{2\ell_0} \cdot p}{\ell_0 + \frac{1}{2}} + \left(1 + \frac{p}{2\ell_0 - 1} - \frac{p(1-p)}{2(2\ell_0 - 1)^2} + \frac{p(1-p)(2-p)}{6(2\ell_0 - 1)^3}\right) - 1 \\
 &\quad \left( \text{since } (1+x)^p < 1 + px - \frac{p(1-p)}{2}x^2 + \frac{p(1-p)(2-p)}{6}x^3 \right. \\
 &\quad \left. \text{according to Lemma 17} \right) \\
 &= \frac{p}{2\ell_0 - 1} \cdot \left( \frac{1}{\ell_0(2\ell_0 + 1)} - \frac{1-p}{2(2\ell_0 - 1)} + \frac{(1-p)(2-p)}{6(2\ell_0 - 1)^2} \right)
 \end{aligned}$$



$$\begin{aligned}
&< \frac{p}{2\ell_0 - 1} \cdot \left( \frac{1}{\ell_0(2\ell_0 + 1)} - \frac{1-p}{2(2\ell_0 - 1)} + \frac{1-p}{3(2\ell_0 - 1)^2} \right) \\
&= \frac{p}{2\ell_0 - 1} \cdot \frac{6(2\ell_0 - 1)^2 - (1-p) \cdot (3\ell_0(2\ell_0 + 1)(2\ell_0 - 1) - 2\ell_0(2\ell_0 + 1))}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2} \\
&= \frac{p}{2\ell_0 - 1} \cdot \frac{6(2\ell_0 - 1)^2 - (1-p)\ell_0 \cdot (12\ell_0^2 - 4\ell_0 - 5)}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2} \\
&< \frac{p}{2\ell_0 - 1} \cdot \frac{6(2\ell_0 - 1)^2 - 2 \cdot (12\ell_0^2 - 4\ell_0 - 5)}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2} \\
&\quad \text{(since } (1-p)\ell_0 > 2 \text{ according to inequality (5)} \\
&\quad \text{and } 12\ell_0^2 - 4\ell_0 - 5 = (6\ell_0 - 5)(2\ell_0 + 1) > 0) \\
&= \frac{p}{2\ell_0 - 1} \cdot \frac{-16\ell_0 + 16}{6\ell_0(2\ell_0 + 1)(2\ell_0 - 1)^2} \\
&< -\eta. \quad \text{(by inequality (7))}
\end{aligned}$$

Thus, for every  $\ell \in [\ell_0 - \frac{1}{2}, \ell_0 + \frac{1}{2}]$ , we have  $f_1(\ell) < -\eta$  and  $f_2(\ell) < -\eta$ . ◀



# Improved Online Load Balancing with Known Makespan

Martin Böhm  

University of Wrocław, Poland

Matej Lieskovský  


Computer Science Institute of Charles University, Faculty of Mathematics and Physics,  
Prague, Czechia

Sören Schmitt  

Department of Mathematics, University of Siegen, Germany

Jiří Sgall  

Computer Science Institute of Charles University, Faculty of Mathematics and Physics,  
Prague, Czechia

Rob van Stee  

Department of Mathematics, University of Siegen, Germany

---

## Abstract

We break the barrier of  $3/2$  for the problem of online load balancing with known makespan, also known as bin stretching. In this problem,  $m$  identical machines and the optimal makespan are given. The load of a machine is the total size of all the jobs assigned to it and the makespan is the maximum load of all the machines. Jobs arrive online and the goal is to assign each job to a machine while staying within a small factor (the competitive ratio) of the optimal makespan.

We present an algorithm that maintains a competitive ratio of  $139/93 < 1.495$  for sufficiently large values of  $m$ , improving the previous bound of  $3/2$ . The value  $3/2$  represents a natural bound for this problem: as long as the online bins are of size at least  $3/2$  of the offline bin, all items that fit at least two times in an offline bin have two nice properties. They fit three times in an online bin and a single such item can be packed together with an item of any size in an online bin. These properties are now both lost, which means that putting even one job on a wrong machine can leave some job unassigned at the end. It also makes it harder to determine good thresholds for the item types. This was one of the main technical issues in getting below  $3/2$ .

The analysis consists of an intricate mixture of size and weight arguments.

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** Online algorithms, bin stretching, bin packing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.10

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.08376> [6]

**Funding** *Martin Böhm:* Research supported by National Science Centre in Poland under grant SONATA 2022/47/D/ST6/02864.

*Matej Lieskovský:* Partially supported by GAUK project 234723, and GA ČR project 24-10306S.

*Jiří Sgall:* Partially supported by GA ČR project 24-10306S.

## 1 Introduction

ONLINE LOAD BALANCING WITH KNOWN MAKESPAN is an online problem defined as follows. At the start of the input, the number  $m$  of machines is revealed, followed by a sequence of jobs with sizes in  $[0, 1]$ , arriving one by one. Each job needs to be assigned to a machine,



© Martin Böhm, Matej Lieskovský, Sören Schmitt, Jiří Sgall, and Rob van Stee;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 10; pp. 10:1–10:21



Leibniz International Proceedings in Informatics

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

and the load of a machine is the total size of the jobs assigned to it. The algorithm is guaranteed a priori that the entire sequence of jobs can be scheduled on  $m$  machines so that the makespan (the load of the most-loaded machine) is at most 1. The objective of the algorithm is to schedule the jobs on the machines as they arrive, minimizing the makespan  $R$  of the online schedule, which is allowed to be larger than 1. The value  $R$  is also known under the name *stretching factor*.

The problem was first introduced in 1998 by Azar and Regev [3, 4] under the name ONLINE BIN STRETCHING, and studied intensively since [8, 9, 15, 17, 21]. Among its given applications is container repacking [7] and reallocation during a server upgrade. This scheduling problem shares its terminology and some algorithmic ideas with ONLINE BIN PACKING. The overarching goal of the research of ONLINE BIN STRETCHING and other related problems over the last few decades is to learn how a small amount of additional knowledge ahead of time (such as knowledge of the makespan) impacts the best possible competitive ratio for the quintessential online problem ONLINE LOAD BALANCING [16].

To that end, another closely related problem is ONLINE LOAD BALANCING WITH KNOWN SUM OF PROCESSING TIMES, where we have a guarantee that the total volume of jobs is at most  $m$ , but the optimum can be larger than 1. (e.g., if jobs larger than 1 appear in the input sequence). For comparison in ONLINE BIN STRETCHING we have a guarantee on the makespan which is stronger, while in the classical ONLINE LOAD BALANCING problem we have no guarantee. Having information on the total volume of jobs or the makespan could be viewed as particular kinds of *advice* given to the online algorithm [10, 11, 24].

To answer the general question above quantitatively, the state of the art is the following. For ONLINE LOAD BALANCING, Fleischer and Wahl [13] presented a deterministic algorithm with competitive ratio approximately 1.92, and Rudin [25] showed that no deterministic algorithm can be better than 1.88-competitive. Kellerer et al. [18] showed that having a guarantee on the sum of processing times allows an approximately 1.585-competitive algorithm as  $m$  goes to infinity, matching the lower bound of Albers and Hellwig [2]. Finally, for ONLINE BIN STRETCHING, Böhm et al. [8] presented an algorithm with stretching factor  $3/2$ , and Azar and Regev [4] showed that no algorithm can have a stretching factor below  $4/3$ .

### Our contribution

We propose an online algorithm for ONLINE BIN STRETCHING that is able to surpass the  $3/2$  threshold:

► **Theorem 1.1.** *For  $m \geq 60000$  and for  $\varepsilon = 1/31$ , there exists an online algorithm for ONLINE BIN STRETCHING with stretching factor  $3/2 - \varepsilon/6 = 139/93 < 1.495$ .*

For  $\varepsilon = 1/62$  the algorithm works already for  $m \geq 3300$ . Our algorithm builds upon the main concepts of its immediate predecessors [15, 8], by keeping a portion of the bins empty until a later phase of the input, and by tracking combinatorial properties of the items using a *weight-based analysis*. Any feasible algorithm must follow this general structure. However, once the stretching factor is set below  $3/2$ , new types of items appear which require great care to pack efficiently. See Figure 1 and the full version. The level of complexity of our algorithm as well as its analysis significantly surpasses the previously best-known results. For instance, it now becomes necessary to use new item types when we start to fill up previously used bins later in the algorithm, as most of the initial item types do not fit well in the remaining space. Achieving a ratio below  $3/2$  for *all* values of  $m$  seems to be much harder still, as we often have constantly many bins which are only half-full; only when  $m$  is large is the number of such bins negligible.

## History and related work

The first results on ONLINE BIN STRETCHING have appeared even before the introductory paper of Azar and Regev in 1998; a year before, Kellerer et al. [19] already discovered the matching lower and upper bound of  $4/3$  for the case  $m = 2$ . Since the beginning, some research works focus on the stretching factor for general number of bins  $m$ , while others focus on the special cases for  $m$  small and fixed. One interesting property of ONLINE BIN STRETCHING with fixed  $m$  is that both the best-known lower bounds [21] and some algorithms [22] were designed using a computer-aided approach based on the *Minimax* algorithm, as initially proposed by Gabay et al. [14].

For any value  $m \geq 2$  a general lower bound of  $4/3$  comes from Azar and Regev [4]. For  $m = 3$ , the best-known algorithm is by Böhm et al. [7]. The remaining lower and upper bounds for the range  $3 \leq m \leq 8$ , listed in the table below, were designed by multiple variants of computer-aided search; the results are by Böhm and Simon [9], Lhomme et al. [21] and Lieskovský [23].

| $m$         | 3         | 4          | 5          | 6          | 7          | 8          | $\geq 9$        |
|-------------|-----------|------------|------------|------------|------------|------------|-----------------|
| Lower bound | 1.365 [9] | 1.357 [9]  | 1.357 [9]  | 1.363 [21] | 1.363 [21] | 1.363 [21] | $1.\bar{3}$ [4] |
| Upper bound | 1.375 [7] | 1.393 [23] | 1.410 [23] | 1.429 [23] | 1.455 [23] | 1.462 [23] | 1.5 [8]         |

For general  $m$ , Böhm et al. [8] presented the so far best algorithm in 2017 which achieves stretching factor  $3/2$ ; this result was preceded by a long sequence of steady improvements on the algorithmic front, among others by Kellerer and Kotov [17] and Gabay et al. [15]. Recently in [20], Lhomme et al. give first results for randomized algorithms. They show that for  $m = 2$  there exists a  $5/4$ -competitive randomized algorithm that outperforms the optimal deterministic algorithm. Furthermore, they provide lower bounds for  $2 \leq m \leq 4$  on the competitive ratio of randomized algorithms.

For some small fixed values of  $m$ , especially  $m = 2$ , also specialized problems related to ONLINE BIN STRETCHING have been investigated previously; for example, Epstein [12] considered online bin stretching with two machines (bins) of uniformly related speed and Akaria and Epstein [1] considered online bin stretching on two bins with grade of service and migration.

## 2 Structure of the algorithm

From now on, as is common in the literature on ONLINE BIN STRETCHING and because we are dealing with a packing problem, we refer to bins, levels of bins and items instead of machines, loads of machines and jobs, respectively. Our initial setting is that the offline optimum bins have size 1 and the bins usable by our algorithm have size  $R = 3/2 - \varepsilon/6$ .

We assume that the number of bins  $m$  is at least 60000. We scale the sizes of the bins such that an offline bin has size 12 and an online bin has size  $18 - 2\varepsilon$ . (We use  $2\varepsilon$  here so that half of the size of an online bin is a more convenient value.) Our goal is to construct an algorithm which works for the largest possible value of  $\varepsilon$ . We will eventually set  $\varepsilon = 1/31$ , but for an easier understanding of the relationships between the various values we will mostly use symbolic calculations. Scaling the offline bin size to 12 allows us to work with near-integer type thresholds, which is convenient. After scaling, the total size of the jobs on input is at most  $12m$ .

Our algorithm uses the algorithms Best Fit and First Fit as subroutines. These algorithms work as follows. Both algorithms open a new bin if the item does not fit into any existing bin. Otherwise, Best Fit places an item in a bin where the item can still fit and that, after

## 10:4 Improved Online Load Balancing with Known Makespan

placement, leaves the least amount of remaining empty space in the bin. This means it uses the most-filled bin that can still accept the item. First Fit always places the item in the first bin in which it will fit, using the order in which it opened the bins. In this paper, we will sometimes fix the order in which the bins are to be used in advance, namely if these bins already contain some items. This means that we are applying First Fit to variable-sized “bins” (the empty spaces in the actual bins). We give a proof for the performance of First Fit on variable-sized bins which may be of independent interest.

► **Lemma 2.1.**<sup>1</sup> *Consider a set  $V$  of bins that is packed by First Fit of which at least the last  $|V| - 2$  bins contain at least  $k$  items. If  $|V| \geq 3$ , the total level of the bins in  $V$  is more than*

$$\frac{k|V|}{k+1}.$$

► **Lemma 2.2.** *For any set of  $v$  variable-sized bins that is packed using First Fit, the following property holds. If at least  $k < v/2$  items are packed into each bin, the total size of all the items packed into these bins is at least*

$$\frac{k}{k+1} \sum_{j=k}^{v-k} s(j),$$

where the size of the  $j$ -th bin is denoted by  $s(j)$ . This even holds if the number of bins increases while First Fit is running (in this case  $v$  is the final number of bins).

**Proof.** Let the bins be sorted by the order of First-Fit.

We look at an  $(k+1)$ -tuple  $(j, j+1, \dots, j+k)$  with  $1 \leq j \leq v-k$ . Let  $\alpha$  be the largest empty space of bins  $j, \dots, j+(k-1)$ . The items in bin  $j+k$  have size at least  $\alpha$ . Bins  $j, \dots, j+(k-1)$  on the other hand are filled to at least  $s(j) - \alpha, \dots, s(j+(k-1)) - \alpha$ . We know that at least  $k$  items of size at least  $\alpha$  are packed in bin  $j+k$ , so in these  $k+1$  bins we have an overall load of at least  $\sum_{i=j}^{j+(k-1)} s(i)$ . Applying this bound for  $j = 1, 2, \dots$  we find guarantees for First-Fit of at least  $\sum_{i=1}^k s(i) + \sum_{i=k+2}^{2k+1} s(i) + \dots, \sum_{i=2}^{k+1} s(i) + \sum_{i=k+3}^{2k+2} s(i) + \dots$ , etc. Adding all these bounds gives

$$(k+1) \cdot FF \geq \sum_{i=1}^{k-1} i \cdot s(i) + \sum_{i=k}^{v-k} k \cdot s(i) + \sum_{i=v-(k-1)}^{v-1} (v-i) s(i) > k \sum_{i=k}^{v-k} s(i). \quad \blacktriangleleft$$

### 2.1 Item types

Our algorithm initially uses the following item types; once we start filling up the bins in the fill-up phase, it will be necessary to use different types because of the amount of space that will be left. We group some item types into supertypes. There are two intervals for small items, as these items are packed the same way. The three weighting functions are related to the three types of items that fit only once in an offline bin. Having three separate such types is a consequence of the existence of quarter items (see Figure 1).

<sup>1</sup> The simple proof of this lemma can be found, e.g., in [5].

| Supertype                 | –                  |                    |                   | middle             |                    |                   | dominant            |     |
|---------------------------|--------------------|--------------------|-------------------|--------------------|--------------------|-------------------|---------------------|-----|
| Type                      | small              | quarter            | small             | nice               | half               | large             | big                 | top |
| Maximum size              | $3 - 3\varepsilon$ | $4 - 4\varepsilon$ | $5 + \varepsilon$ | $6 - 2\varepsilon$ | $6 + 2\varepsilon$ | $9 - \varepsilon$ | $10 + 6\varepsilon$ | 12  |
| Weight $w_{\text{top}}$   | 0                  | 1                  | 1                 | 2                  | 2                  | 2                 | 3                   | 4   |
| Weight $w_{\text{big}}$   | 0                  | 0                  | 0                 | 2                  | 2                  | 2                 | 4                   | 4   |
| Weight $w_{\text{large}}$ | 0                  | 0                  | 0                 | 0                  | 2                  | 4                 | 4                   | 4   |

We describe the ideas behind these type thresholds in detail in the full version of the paper. Here we describe some fundamental properties of the various (super-)types. See also Figure 1.

Dominant items fit only once in an online bin. Nice items fit twice in an offline bin and can be placed in an online bin while still leaving room for another item of any size. (These items are indeed in principle nice to pack, but we still need to be very careful with them.) Half and large items fit twice in an online bin, but a large item cannot be packed together with a half item in an offline bin (due to the thresholds  $6 - 2\varepsilon$  and  $6 + 2\varepsilon$ ), whereas two half items *may* fit together in an offline bin.

In our algorithm, we will pack small items only to a level of  $6 - 6\varepsilon$  at the beginning to leave room for one top item or two half items. As described above, using First Fit guarantees that more than  $4 - 4\varepsilon$  is packed in almost all bins that are packed like this. Of course we get the same guarantee for small items of size more than  $4 - 4\varepsilon$ , and this is what motivates the upper bound  $4 - 4\varepsilon$  for quarter items. It is also the same guarantee that we will achieve on average for (a certain subset of) the bins with quarter items (some bins will contain two quarter items). A big item fits in an online bin with two quarter items, and this is the reason that the dominant items are divided into two types.

► **Definition 2.3.** For a partial input  $I_{\text{partial}}$ , let the value  $\text{TOPTHREAT}$  (resp.,  $\text{BIGTHREAT}$ ,  $\text{LARGETHREAT}$ ) be the maximum number of top items (resp., big items, large items) in  $I_{\text{future}}$  so that  $I_{\text{partial}} \cup I_{\text{future}}$  can be packed in  $m$  bins of size 12, and let  $\text{TOPBLOCK}$  (resp.,  $\text{BIGBLOCK}$ ,  $\text{LARGEBLOCK}$ ) be the set of bins that contain more than  $6 - 2\varepsilon$  (resp.,  $18 - 2\varepsilon - (10 + 6\varepsilon) = 8 - 8\varepsilon, 9 - \varepsilon$ ).

For any packing of a partial input, we have  $\text{TOPTHREAT} \leq \text{BIGTHREAT} \leq \text{LARGETHREAT}$  and  $\text{LARGEBLOCK} \leq \text{BIGBLOCK} \leq \text{TOPBLOCK}$ .

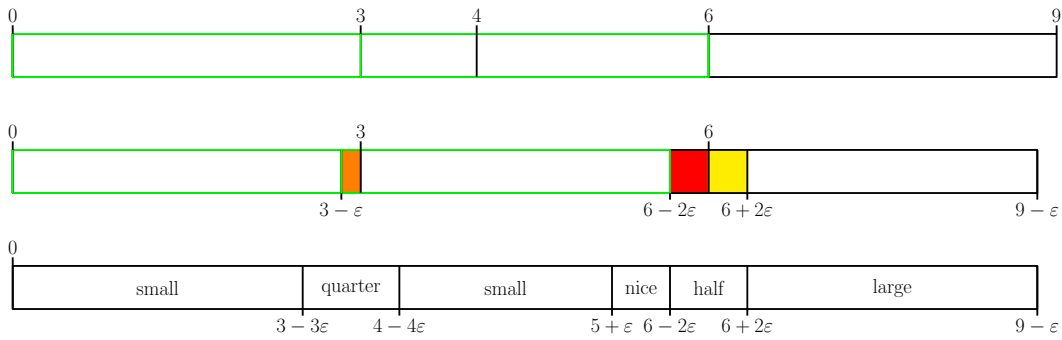
► **Lemma 2.4.** For any feasible input  $I$  and weighting function  $w \in \{w_{\text{top}}, w_{\text{big}}, w_{\text{large}}\}$ , we have  $w(I) \leq 4m$ . For any  $k \geq 0$  and any partial input  $I_{\text{partial}}$ :

- if  $w_{\text{top}}(I_{\text{partial}}) \geq 4k$ , then  $\text{TOPTHREAT} \leq m - k$ ,
- if  $w_{\text{big}}(I_{\text{partial}}) \geq 4k$ , then  $\text{BIGTHREAT} \leq m - k$ ,
- if  $w_{\text{large}}(I_{\text{partial}}) \geq 4k$ , then  $\text{LARGETHREAT} \leq m - k$ .

**Proof.** The bound  $w_{\text{large}}(I) \leq 4m$  follows from the type thresholds (a large and a half item do not fit together in an offline bin). For the other two weighting functions, note that for an item  $i$  of type  $j$ , the weight  $w_{\text{top}}(i) = \lfloor \frac{5}{12}s_j \rfloor$  and  $w_{\text{big}}(i) = 2(\lfloor \frac{3}{12}s_j \rfloor)$ , where  $s_j$  is the infimum size of an item of type  $j$  (where the small items are split into two separate types for this calculation, one for each range of small items). Intuitively,  $w_{\text{top}}$  counts the number of items larger than  $\frac{12}{5}$ , that is, items that fit at most four times in an offline bin. Similarly,  $w_{\text{big}}$  counts items larger than  $\frac{12}{3}$ , and multiplies the result by two. The bounds  $w_{\text{top}}(I) \leq 4m$  and  $w_{\text{big}}(I) \leq 4m$  follow. ◀

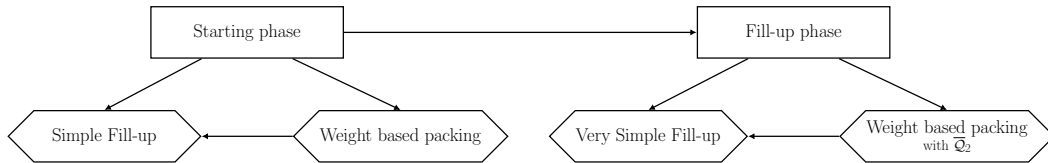
The following invariant is a necessary property of any feasible algorithm and we will maintain it and other invariants throughout the processing of the input.

## 10:6 Improved Online Load Balancing with Known Makespan



■ **Figure 1** (Sketch, using  $\varepsilon = 1/6$ ) A comparison of the important thresholds for an algorithm with competitive ratio  $3/2$  (top) and an algorithm with competitive ratio strictly less than  $3/2$  (middle). The thresholds our algorithm uses are displayed at the bottom. The green box indicates (half) the difference between the online and offline bin size. In the top figure, 6 is also a point where the amounts that can be packed in a bin change, both online and offline.

We immediately see that in the middle figure items exist which did not exist before (red); for a competitive ratio of  $3/2$ , the online algorithm can pack more items per bin for *all* items smaller than 9. Moreover, items in the orange range can block some items of maximal size from being packed in the same bin, if we pack two such items in one bin. Finally, the fact that the red range exists means that items just larger than this (yellow) also need to be packed more carefully than before.



■ **Figure 2** An overview of the phases and states.

► **Invariant 2.5.** We have  $TOPTHREAT \leq m - TOPBLOCK$  and  $BIGTHREAT \leq m - BIGBLOCK$ .

We will *not* be able to maintain  $LARGETHREAT \leq m - LARGEBLOCK$  throughout the algorithm (not even in the starting phase). However, fortunately large items can be placed twice in an online bin. Since these items can have size up to  $9 - \varepsilon$ , a bin must be completely empty in order to guarantee that two large items may be packed in it.

### 2.2 Phases and states

In the starting phase, we use bins one by one, while staying below a level of  $6 - 2\varepsilon$  unless there is a very good reason not to do so. If many relatively large items arrive, we may reach a state where it is sufficient to use First Fit for all remaining items (Simple Fill-Up) or where we know by weight that all items can be packed (Weight-based packing). Otherwise, we will eventually go to the Fill-up phase, where we start filling up the bins that previously received less than  $6 - 2\varepsilon$  (or up to  $8 - 8\varepsilon$  in the case of bins with two quarter items). In this phase we will eventually also reach a state where we know that the remaining input can be packed, either by size or by a weight argument.



### The starting phase

From the lens of a single bin, our algorithm typically either packs items until a bin is full – which is typical for bins containing a single item type, such as the middle items – or it packs them only up to a level of  $6 - 6\epsilon$ , particularly for items of size at most  $6 - 6\epsilon$ .

However, as we have already seen, quarter items do not fit in this framework. On the one hand, we need to avoid packing many quarter items alone in bins (bad packing guarantee) while on the other hand, we also cannot pack too many quarter items in pairs in bins that do not yet contain anything else: that could block top items from being packed (Invariant 2.5 would be violated).

Ideally, we would like to pack items as follows:

- top items or pairs of half items with small items
- big items with quarter items
- large items in pairs
- nice items three per bin

In this way, all bins would have a weight of at least 4 in  $w_{\text{top}}$  and  $w_{\text{big}}$  and they would also all be more than 12 full (except for bins that contain one big item and one quarter item and bins that contain a top item/two half items smaller than 6 and not enough small items). There are several problems in using these methods, however:

- For bins that are planned to contain items of two different types, or two items of one type, it is not known whether the second type or item will ever arrive.
- Packing large items and smaller middle items into separate bins can easily lead to instances that cannot be packed (if there are two bins with single middle items that fit together in an offline bin, and then many top items arrive).

We can work around the first problem by changing our packing methods after a certain number of bins have received items of only one type, in particular if many small or quarter items arrive. Basically, our algorithm will first aim to reach the ideal packing described above. When sufficient volume has been packed, we go back and start filling up the already used bins. This is the fill-up phase of our algorithm.

The second problem requires us to be very careful with nice items in particular, since some nice items fit with some large items in an offline bin. Packing nice items three per bin in dedicated bins will be fine. However, we cannot afford to do this already starting from the very first bin with nice items, as there could also be a bin with one half item and another bin with one large item at the same time, blocking too many bins for top items so that the algorithm fails. Fortunately, a bin with only a nice item can still receive an item of any other type, so we will pack one nice item alone before starting to pack them three per bin from the second bin onwards. We still need to be very careful if both half and nice items arrive.

### Good situations and the fill-up phase

We may be fortunate and reach a situation where many bins are filled to (significantly) more than 12. In this case it will be sufficient to pack the remaining items by essentially using First Fit. This is one example of a *good situation*. This is our term for a configuration which ensures that all remaining items can be packed, usually by using a very simple algorithm. This one is called the **First Fit case**.

It may also happen that many relatively large items arrive early. In this case we may reach a state where we know that a small or quarter item will never need to be packed into an empty bin anymore, because they are packed in existing bins first and we would reach the

First Fit case before using an empty bin. To ensure that the algorithm does succeed in all cases, even if all bins receive items, we will always use Best Fit as last resort for any item (after exhausting all other rules and all empty bins). We call this the **Rule of last resort**.

If many bins contain items but not enough of them contain a total size of more than 12 or sufficiently large items, it becomes important whether there exist bins that contain only small items or only single quarter items. If that is the case, we will go to the fill-up phase, in which we start filling up the nonempty bins using different item types. Otherwise, we will remain in the starting phase and we will eventually reach a good situation or the input will end.

### The $(9 - \varepsilon)$ -guarantee

We need to determine when exactly it is safe to start filling up bins in which we have already packed some items, without failing for instance to the threat of top items. To be precise, once we start filling up bins, we need a guarantee that this *remains* feasible no matter what the remaining input is. This will certainly require us to pack a sufficient total size in each bin that we fill up, as we always need to maintain Invariant 2.5.

Our cutoff for starting to fill up bins will be the point at which we know for certain that the future number of *big* items is (and will remain!) strictly smaller than the number of bins in which big items can still be packed (so,  $\text{BIGTHREAT} < m - \text{BIGBLOCK}$ ). There are in principle two ways by which we can know this: by considering weight and by considering volume. The problem with using a weight-based guarantee is that for instance small items can start arriving, which do not have weight. If we start filling up bins using small items, we can soon reach a point where the weight-based bound for  $\text{BIGTHREAT}$  has not changed, but  $\text{BIGBLOCK}$  has increased and we fail when many big items arrive.

We therefore use a volume-based bound. We need to be careful also here. Suppose that already  $2m/3$  bins contain small items, and each such bin has a level in the range  $(4 - 4\varepsilon, 6 - 6\varepsilon]$ . Now suppose that many big items start arriving one by one. These big items do *not* bring us really closer to the point where we can safely start filling up the nonempty bins, because every time that we pack a big item  $\text{BIGTHREAT}$  decreases by 1 and  $m - \text{BIGBLOCK}$  decreases by 1. Similarly, top items bring us only slowly closer to this point (since they are slightly larger than big items).

We will start the fill-up phase once we know the so-called  **$(9 - \varepsilon)$ -guarantee** holds:

Whenever new items of total size  $9 - \varepsilon$  arrive,  $\text{BIGTHREAT}$  decreases by at least 1.

Having the  $(9 - \varepsilon)$ -guarantee essentially ensures that packing  $9 - \varepsilon$  per bin is sufficient to maintain Invariant 2.5, although the problem of  $m$  large items arriving remains and needs to be dealt with separately. Maintaining this average is not at all straightforward, since we also have to make sure not to use too many *empty* bins too early, in order to pack as many pairs of large items into them as possible.

We present a very careful method of filling the nonempty bins which takes care to use the remaining space in those bins as efficiently as possible, using new item types which are tailored to the remaining space. This method consists of several stages.

## 2.3 Bin types

During the execution of the algorithm, each bin in the instance will be assigned a specific type. Sets of bins of a certain (sub)type are denoted typically by script letters (possibly with an index). We define six main types of bins. We use the corresponding lower case letters to refer to numbers of bins of a type: for instance,  $\ell = |\mathcal{L}|$  and  $\delta = |\Delta|$ .

- $\mathcal{E}$  Empty bins.
- $\mathcal{L}$  Large-complete bins. This is a set of bins that reduce `LARGETHREAT`; more generally, they reduce the number of items with weight that can still arrive. Specifically, the number of large items that can still arrive will always be at most  $m - \ell$ , and the total weight that can still arrive will be at most  $4(m - \ell)$ . Since items without weight can still arrive however, and bins in  $\mathcal{L}$  do not necessarily contain at least 12, large-complete bins do still accept items. A formal definition of these bins follows below.
- $\mathcal{S}$  Bins with only *small* items. At most  $6 - 6\varepsilon$  of small items is packed in each such bin.
- $\Delta$  At most four bins that contain two nice items or a single middle item and maybe some other items. See below for more details.
- $\mathcal{Q}$  Bins that are not in  $\Delta$  in which the first item (or the second, if the bin was previously in  $\Delta$ ) is a quarter item and that are unmatched. (The algorithm sometimes matches some bins in  $\mathcal{Q}$ ; these bins are then moved to a special subset  $\mathcal{Q}_{\text{match}}$ .)
- $\mathcal{N}$  Bins in which the first two items are nice items and the third item is nice or half.

Bins started by nice items are filled to triples of nice items in the ideal packing and kept separate to achieve this; these bins can become large-complete upon receiving a dominant item. With this large-scale picture in mind, the large-complete bins are defined as follows. These bins require a careful definition because nice items may exist.

► **Definition 2.6.** *A bin is called large-complete if it satisfies all of the following conditions.*

- *it has  $w_{\text{big}} \geq 4$ ,*
- *it contains an item larger than 6 or two items larger than  $6 - 2\varepsilon$ ,*
- *the bin was never in  $\mathcal{Q}$ .*

It can be seen that each bin with  $w_{\text{big}} \geq 4$  has a big item or  $w_{\text{top}} \geq 4$ . A large-complete bin does not necessarily contain a large item or a dominant item. The first condition ensures that these bins contain as much weight as any offline bin. The second condition implies that `LARGETHREAT`  $\leq m - \ell$  at all times. Note that this does not follow from the first condition alone, as a bin could contain two nice items, and a nice and a large item may fit together in an offline bin.

The set  $\Delta$  contains at most four exceptional bins used for careful handling of middle items. Each of these bins will be created explicitly in our algorithm if they are needed. There are the following four kinds of bins in  $\Delta$ .

- $\Delta_{\text{large}}$  one bin that contains a single large item, nothing else.
- $\Delta_{\text{half}}$  one bin that contains a single half item and possibly small items of total size at most  $6 - 6\varepsilon$  (notation  $\Delta_{\text{half}}^S$ ) or a quarter item (notation  $\Delta_{\text{half}}^Q$ ), nothing else. If the bin contains *only* a half item we call it  $\bar{\Delta}_{\text{half}}$ , else  $\Delta_{\text{half}}^+$ .
- $\Delta_{\text{nice},1}$  at most two bins that contain one nice item and nothing else.
- $\Delta_{\text{nice},2}$  one bin that contains two nice items and nothing else.

Our algorithm will use the so-called **nice rule** as long as possible: do not pack nice items into  $\Delta_{\text{large}} \cup \Delta_{\text{half}}$  and do not pack half or large items into  $\Delta_{\text{nice},1}$ . This rule ensures that nice items get packed into dedicated bins as much as possible (three per bin) so that we gain on these items (both by weight and by packed size per bin) compared to the optimal packing. This in turn ensures that nice items will hardly occur in inputs that are important for our analysis; see the weighting function  $w_{\text{large}}$ .

► **Definition 2.7.** *A bin is in  $\mathcal{Q}$  if it satisfies the following properties:*

- *The first item is a quarter item,*
- *The bin is not in  $\Delta_{\text{half}}$ ,*
- *The bin has not been matched (see algorithm).*

## 10:10 Improved Online Load Balancing with Known Makespan

Additionally, a bin that was in  $\overline{\Delta}_{\text{half}}$  and then received a quarter item and finally another half or larger item is also in  $\mathcal{Q}$  as long as it has not been matched.

We define the subset of  $\mathcal{Q}$  of bins in which the first two items are quarter items by  $\mathcal{Q}_2$ , and  $\mathcal{Q}_1 := \mathcal{Q} \setminus \mathcal{Q}_2$ . A bin in  $\mathcal{Q}_1$  may leave  $\mathcal{Q}$  (and  $\mathcal{Q}_1$ ) by receiving a half item (it enters  $\Delta_{\text{half}}$ ); such a bin may later rejoin  $\mathcal{Q}$  by receiving a half or larger item. Bins in  $\mathcal{Q}$  may also leave  $\mathcal{Q}$  permanently by being matched (two bins in  $\mathcal{Q}_1$  to one bin in  $\mathcal{Q}_2$ ).

Instead of using the partition  $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ , we will also consider the useful partition of  $\mathcal{Q}$  in the table below. (Some of these subsets may be empty.)

| Bin type                     | Conditions on contents  |
|------------------------------|---|
| $\overline{\mathcal{Q}}_1$   | A single quarter item, nothing else   |
| $\overline{\mathcal{Q}}_2$   | Two quarter items, nothing else   |
| $\mathcal{Q}_{1,\text{big}}$ | First item is a quarter item, second item is big  |
| $\mathcal{Q}_5$              | First item is a quarter item, $w_{\text{big}} \geq 4$ , bin is not in $\mathcal{Q}_{1,\text{big}}$<br>Or: The first three items are (in this order) half, quarter, half or larger |

We let  $\overline{\mathcal{Q}} = \overline{\mathcal{Q}}_1 \cup \overline{\mathcal{Q}}_2$ . Bins in  $\mathcal{Q}_5$  can be in  $\mathcal{Q}_1$  (for instance, bins with a top item) or in  $\mathcal{Q}_2$ ; we keep track of their membership via the sets  $\mathcal{Q}_{1,5} := \mathcal{Q}_5 \cap \mathcal{Q}_1$  and  $\mathcal{Q}_{2,5} := \mathcal{Q}_5 \cap \mathcal{Q}_2$ . All bins in  $\mathcal{Q}_5$  will have  $w_{\text{top}}$ -weight 5 (or more), explaining the name  $\mathcal{Q}_5$ . For comparison, bins in  $\mathcal{Q}_1$  have  $w_{\text{top}}$ -weight at least 1 and bins in  $\mathcal{Q}_2$  have  $w_{\text{top}}$ -weight at least 2.

Bins in  $\mathcal{Q}_{1,\text{big}}$  may get matched (pairwise) to bins in  $\mathcal{Q}_{2,5}$ ; this is explained in the algorithm (Step 3). The set of matched bins is denoted by  $\mathcal{Q}_{\text{match}}$ .

Since the first two items in each bin in  $\mathcal{Q}_2$  are quarter items, we have  $\mathcal{Q}_2 \cap \mathcal{Q}_{1,\text{big}} = \emptyset$ . We use the membership of bins in  $\mathcal{Q}_1$  and  $\mathcal{Q}_2$  to keep track of the distribution of quarter items in the non-large-complete bins. In our proofs, we will assign weight from the quarter items in  $\mathcal{Q}_1$  to bins in  $\mathcal{Q}_2$  on the one hand (so we need sufficiently many bins in  $\mathcal{Q}_1$ ) and assign volume from bins in  $\mathcal{Q}_2$  to bins in  $\mathcal{Q}_1$  on the other hand (so we need sufficiently many bins in  $\mathcal{Q}_2$ ). The separation from large-complete bins and the separation of  $\mathcal{Q}_{1,5}$  and  $\mathcal{Q}_{2,5}$  will help us maintain an almost fixed ratio  $q_1 : q_2$ . Because of various half-full bins, we will need some additional bins in  $\mathcal{Q}_1$  (at most 15 in the fill-up phase) before starting to create bins in  $\mathcal{Q}_2$ . A bin in  $\overline{\mathcal{Q}}_1$  that receives a half item leaves  $\mathcal{Q}$  and enters  $\Delta_{\text{half}}$  (and  $\Delta_{\text{half}}^{\mathcal{Q}}$ ). If it later receives another half item or a large, it returns to  $\mathcal{Q}$ , namely  $\mathcal{Q}_{1,5}$ , or moves to  $\mathcal{L}$ . Summarizing, we have the following disjoint unions.

$$\mathcal{Q}_1 = \overline{\mathcal{Q}}_1 \cup \mathcal{Q}_{1,\text{big}} \cup \mathcal{Q}_{1,5} \tag{1}$$

$$\mathcal{Q}_2 = \overline{\mathcal{Q}}_2 \cup \mathcal{Q}_{2,5}. \tag{2}$$

We define the set of *complete* bins  $\mathcal{C}$  as the set of bins that from the point of view of the algorithm (and the analysis) do not need to receive any specific items, as follows:

$$\mathcal{C} := \mathcal{L} \cup \mathcal{Q}_5 \cup \mathcal{Q}_{\text{match}} \cup \mathcal{N}.$$

Into these bins, any item may be packed. Finally, the unmatched nonempty bins that are not large-complete are called *regular* (set  $\mathcal{R}$ ). We have

$$\mathcal{R} = \mathcal{S} \cup \mathcal{Q} \cup \mathcal{N} \cup \Delta = \mathcal{S} \cup \overline{\mathcal{Q}}_1 \cup \overline{\mathcal{Q}}_2 \cup \mathcal{Q}_{1,\text{big}} \cup \mathcal{Q}_5 \cup \mathcal{N} \cup \Delta. \tag{3}$$

At all times, each bin is in exactly one of the sets  $\mathcal{R}, \mathcal{Q}_{\text{match}}, \mathcal{L}, \mathcal{E}$ .

We will show eventually that in the starting phase, some bins remain empty or we can guarantee that all remaining items can be packed (possibly using different methods). However, the partitioning of the sets shown here remains valid even after we run out of empty bins

apart from the fact that some bins may contain some items that do not belong there; for instance, there could be some small items packed into  $\overline{Q}_2$ . Also, after we run out of empty bins the nice rule may be violated. We will maintain the following invariant.

► **Invariant 2.8.** *There is never a bin in  $\overline{Q}_1 \cup \overline{Q}_2$  at the same time as a bin with a big item as its only item with weight.*

► **Invariant 2.9.** *We have  $LARGETHREAT \leq m - (c - n)$  as long as the nice rule is followed.*

► **Lemma 2.10.** *Invariant 2.9 holds for any packing of items.*

**Proof.** As long as the nice rule is followed, all complete bins except for the ones in  $\mathcal{N}$  contain two half items or an item larger than 6 and have  $w_{\text{big}} \geq 4$ . ◀

From this bound it can be seen that the possible existence of bins in  $\mathcal{N}$  force us to keep bins empty for *pairs* of large items, since we cannot ensure  $LARGETHREAT \leq m - c$ .

## 2.4 Proof overview

The present version omits essentially all of the proofs. Here we merely give an overview. The proof begins with some initial observations regarding how many bins there can be of different types and how much they contain. We then focus on the set  $\mathcal{Q}$  and prove that up to an additive constant,  $2q_2 = q_1$  throughout the starting phase. The (almost) fixed ratio  $q_2 : q_1$  is used to help show Invariant 2.5 for top items and to show a packing guarantee for  $\mathcal{Q}$ . There will be constantly many bins that do not satisfy our packing guarantees, these bins will be in a set  $\mathcal{X}$ .

In the starting phase, either some bins remain empty,  $\overline{q}_2 > 0$ , or all items get packed. It turns out that Invariant 2.5 is maintained as long as we do not use the rule of last resort (essentially, as long as some bins are empty). There exist so-called good situations in which we can guarantee that all remaining items can be packed (possibly using a different algorithm). We show that Invariant 2.5 is maintained in the entire starting phase or we reach a good situation. More generally, the algorithm does not fail in the starting phase. We find that packing  $9 - \varepsilon$  additionally in each non-complete bin in the fill-up phase is enough to maintain Invariant 2.5 in the fill-up phase as well.

To analyze the fill-up phase, we first consider some simple cases (essentially, new good situations). We then continue by showing that the algorithm does not fail in the first three stages of the fill-up phase. Linear programs are used to show that the algorithm does not fail in the fill-up phase.

### 3 Algorithm in the starting phase

Whenever the algorithm uses or attempts to use a set  $A$  to pack an item in the following description, we use First Fit on the bins in  $A$ , unless otherwise stated. The notation  $A \rightarrow B$  means that a bin in the set  $A$  moves to  $B$  by receiving an item of the current type.

**Step 1: Using and creating complete bins** Try the following in this order.

- for half and large items:  $\Delta_{\text{nice},2} \rightarrow \mathcal{N} \subseteq \mathcal{C}$ ,
- for non-small items:  $\mathcal{Q}_{1,\text{big}} \rightarrow \mathcal{Q}_{1,5} \subseteq \mathcal{C}$
- use a complete bin (a bin in  $\mathcal{C} = \mathcal{L} \cup \mathcal{Q}_5 \cup \mathcal{Q}_{\text{match}} \cup \mathcal{N}$ ).

## 10:12 Improved Online Load Balancing with Known Makespan

- create a complete bin if this does not violate the nice rule (page 9).  
First try the bins  $\overline{Q}_2, \overline{Q}_1, \Delta_{\text{half}}^Q$  in this order. Among other bins, use Best Fit to create a bin in  $\mathcal{L}$ , but do not pack a half item into  $\Delta_{\text{large}}$  (yet).<sup>2</sup>

**Step 2: Packing rules for each item type** If an item is not packed yet, we apply the following rules depending on the item type.

**Small:** First Fit on bins in  $\mathcal{S} \cup \Delta_{\text{half}}^S$  while packing at most  $6 - 6\varepsilon$  of small items in each bin,  $\overline{\Delta}_{\text{half}} \rightarrow \Delta_{\text{half}}^S, \mathcal{E} \rightarrow \mathcal{S}$ .

**Quarter:** If  $|\mathcal{Q}_1| + \delta_{\text{half}}^Q \geq 2|\mathcal{Q}_2| + 15$  then  $\overline{Q}_1 \rightarrow \overline{Q}_2$ , else  $\overline{\Delta}_{\text{half}} \rightarrow \Delta_{\text{half}}^Q, \mathcal{E} \rightarrow \overline{Q}_1$ .

**Nice:**  $\Delta_{\text{nice},2} \rightarrow \mathcal{N}$ , if  $\delta_{\text{nice},1} = 2$  then  $\Delta_{\text{nice},1} \rightarrow \Delta_{\text{nice},2}, \mathcal{E} \rightarrow \Delta_{\text{nice},1}$ .

**Half:** Best Fit on bins in  $\mathcal{S} \cup \overline{Q}_1 \rightarrow \Delta_{\text{half}}, \Delta_{\text{large}} \rightarrow \mathcal{L}, \mathcal{E} \rightarrow \Delta_{\text{half}}$ .

**Large:**  $\mathcal{E} \rightarrow \Delta_{\text{large}}$ .

**Dominant:** Always packed in Step 1.

**Rule of last resort** If some item cannot be packed according to these rules, which can only happen after we run out of empty bins, we use Best Fit for this item, except that we still follow the nice rule as long as possible. If the nice rule has already been violated, we simply use Best Fit. For future items we still use the packing rules above first.

**Step 3: Matching rule** This step minimizes the number of bins in  $\mathcal{Q}_{1,\text{big}}$ .

If  $|\mathcal{Q}_{1,\text{big}}| \geq 2$  and there is a bin in  $\mathcal{Q}_{2,5}$ , two bins in  $\mathcal{Q}_{1,\text{big}}$  are matched to a bin in  $\mathcal{Q}_{2,5}$  and all three bins are moved from  $\mathcal{Q}$  to  $\mathcal{Q}_{\text{match}}$ .

**Step 4: Swapping rule** Each time that a new bin  $\bar{b}$  in  $\overline{Q}_1$  is created, if there exists a large-complete bin  $b$  with a big item but no other items with weight (such a bin must contain also other items, or we would not have created  $\bar{b}$ ), we virtually swap some items. That is, the bin  $\bar{b}$  is treated as a bin in  $\mathcal{S}$  from now on, and the bin  $b$  supposedly contains a big item and a quarter item. The quarter item is not considered to be the first item in  $b$ , so  $b$  is not in  $\mathcal{Q}$ . This ensures that Invariant 2.8 is maintained.

The swapping rule ensures that big items can be packed together with small items without violating Invariant 2.8 even if quarter items arrive later. Whenever the swapping rule is applied on two bins  $\bar{b} \in \overline{Q}_1$  and  $b \in \mathcal{L}$ , the total size packed into these bins is more than  $18 - 2\varepsilon$  at this point (else  $\bar{b}$  would not have been opened). If the bin  $\bar{b}$  contains less than  $4 - 4\varepsilon$  we reassign volume such that the bin  $\bar{b}$  ends up with exactly  $4 - 4\varepsilon$ . We see that more than  $18 - 2\varepsilon - (4 - 4\varepsilon) = 14 + 2\varepsilon$  remains for the bin  $b$ . This just means that  $\bar{b}$  possibly has slightly more space for additional items than the algorithm calculates with (because it views  $\bar{b}$  as containing the small items that were in  $b$ ).

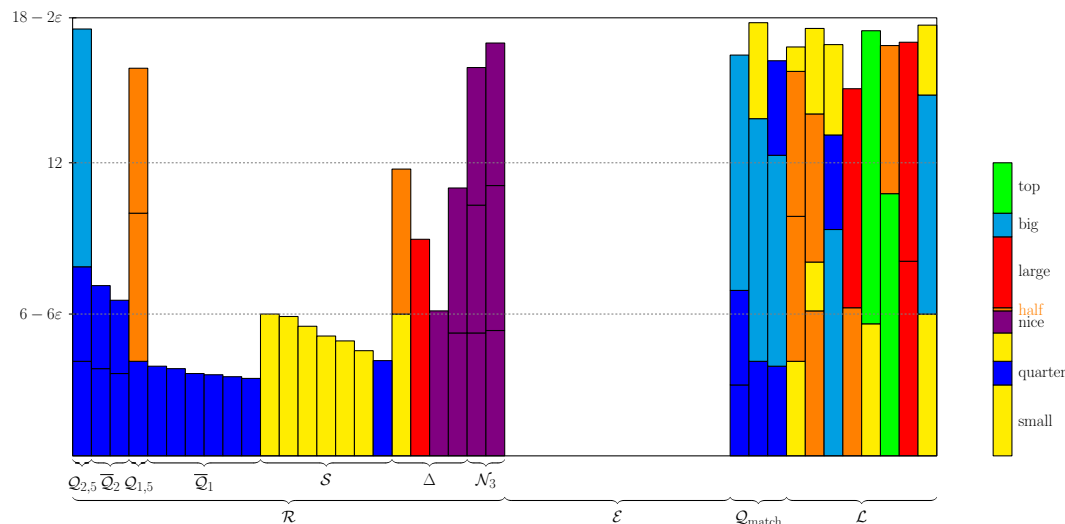
The large-complete bins used by the swapping rule differ from the other bins in  $\mathcal{L}$  only in that they are not used to pack any future item. That is, we ignore such bins in Step 1 (this is not written explicitly in the algorithm; it seemed cleaner to explain this here).

### Transitioning to the fill-up phase

In general, the transition from the starting phase to the fill-up phase happens once the  $(9 - \varepsilon)$ -guarantee starts to hold. This is roughly speaking after packing an average of  $3 + \varepsilon$  on the (non-complete) regular and empty bins and 12 on the complete bins. More precisely

<sup>2</sup> If  $\mathcal{S} \cup \overline{Q}_1 \neq \emptyset$ , we prefer packing half items there rather than in  $\Delta_{\text{large}}$ , because this improves certain packing guarantees. (If  $\mathcal{S} \cup \overline{Q}_1 \neq \emptyset$ ,  $\Delta_{\text{half}}$  exists and a large item arrives, we will have that  $\Delta_{\text{half}} = \Delta_{\text{half}}^S$  which already improves the guarantee.)

after packing  $T_1 := (3 + \varepsilon)m + (9 - \varepsilon)(c + q_{1,\text{big}} + 14)$ . We can guarantee that the algorithm keeps at least roughly  $\mathcal{E}_0 := \frac{1-5\varepsilon}{4-4\varepsilon}(m - \ell - q_{\text{match}}) + \frac{2+8\varepsilon}{4-4\varepsilon}n + \frac{1-5\varepsilon}{4-4\varepsilon}q_{\text{match}} + \frac{4\varepsilon}{4-4\varepsilon}\ell - 48$  bins empty when reaching the packing threshold. For more details see the full version.



**Figure 3** (Sketch) Overview of bins in the starting phase. The three bins in  $\mathcal{Q}_{\text{match}}$  were moved there by the matching rule. The second quarter item in the rightmost bin in  $\mathcal{Q}_{\text{match}}$  arrived there when the bins were already in  $\mathcal{Q}_{\text{match}}$ . The swapping rule was applied to the rightmost bin in  $\mathcal{S}$  and the rightmost bin in  $\mathcal{L}$ . The small items on top of the big item in the rightmost bin arrived before the swapping rule was applied. For visual clarity we have left out a number of bins in  $\overline{\mathcal{Q}}_1$ .

## 4 The fill-up phase

### 4.1 Preliminaries

Once the fill-up phase is reached we refer to the bins by their type they had when the fill-up phase was started and no longer update these sets. E.g., a bin in  $\overline{\mathcal{Q}}_2$  at the start of the fill-up phase that receives a big item in the fill-up phase does not become a bin in  $\mathcal{Q}_{2,5}$  but is referred to as a bin in  $\overline{\mathcal{Q}}_2$  even after receiving the big item. We assume all bins in  $\mathcal{S}$  contain more than  $4 - 4\varepsilon$ , overestimating its total content by at most  $2 \cdot (4 - 4\varepsilon)$ . The bin in  $\Delta_{\text{large}}$  is assumed to contain a  $\text{large}^+$  item once we enter this phase, overestimating its content by at most  $3 + \varepsilon$ . The bin  $\Delta_{\text{half}}^+$  contains an easy item which matches the packing rules in the fill-up phase.

There are three possible states when entering the fill-up phase:

- $s + \overline{q}_1 > 0$
- $s + \overline{q}_1 + \overline{q}_2 = 0$
- $s + \overline{q}_1 = 0$  and  $\overline{q}_2 > 0$

The first case is what we will call the standard case where  $e \geq \mathcal{E}_0 = \Omega(m)$  holds, for which we can guarantee that when using our packing rules we will eventually end in a good situation or the input ends. For the second and third case we can guarantee that we are already in good situations that do not have requirements on  $e$  or  $r$ . For more details we refer to the full version.

► **Definition 4.1.** Let  $e_0$  be the number of empty bins at the start of the fill-up phase.

## 10:14 Improved Online Load Balancing with Known Makespan

For the fill-up phase, we introduce the set  $\mathcal{U}$  of *unused* bins. These are mostly bins that have not received items in the fill-up phase but that we do plan to use for items. At the start of the fill-up phase, these are the bins that are not in  $\mathcal{L} \cup \mathcal{Q}_{\text{match}}$ , so  $|\mathcal{U}| = m - \ell - q_{\text{match}} = r + e_0$ .

Bins in  $\mathcal{N}$  are also not used for items anymore, but are initially counted as part of  $\mathcal{U}$  so that  $\text{LARGETHREAT} \leq u$  (see Invariant 2.9). Bins in  $\mathcal{Q}_5$  are initially in  $\mathcal{U}$  to maintain the proper ratio  $q_1 : q_2$ . At the start of the fill-up phase, the bins in  $\mathcal{U}$  are sorted from left to right. We use the ordering<sup>3</sup>

$$\mathcal{N}, \mathcal{Q}_{2,5}, \overline{\mathcal{Q}}_2, \mathcal{S}, \mathcal{Q}_1, \mathcal{E}, \overline{\Delta}_{\text{half}} \cup \Delta_{\text{large}}$$

where the subsets  $\mathcal{S}$  and  $\mathcal{Q}_1$  are ordered by non-increasing levels and  $\Delta_{\text{half}}^+$  is placed among them if it exists. Indeed, the entire set  $\mathcal{U}$  is essentially sorted by the levels of small and quarter items at the start of the fill-up phase, so for instance bins in  $\mathcal{Q}_1$  (including bins in  $\mathcal{Q}_{1,5}$  and  $\mathcal{Q}_{1,\text{big}}$ ) have level at most  $4 - 4\varepsilon$  for the sorting. Throughout the fill-up phase, by the level of a bin in  $\mathcal{Q}$  we will always mean the total size of the quarter items in this bin at the end of the starting phase. Regarding  $\mathcal{N}$ , it is often convenient to divide the contents of these bins in a part of size at least  $6 - 6\varepsilon$  and a part of size exactly  $9 - \varepsilon$  (and this is why these bins are first in the ordering).

During the fill-up phase, we will maintain a set  $\mathcal{D}$  such that  $\text{TOPTHREAT} \leq u - d$  will hold throughout the fill-up phase. We define a specific initial set  $\mathcal{D}$  below and we will update this set throughout, using the following rules.

**Rule 1** Each bin that is used (in particular bins in  $\mathcal{D}$ ) will receive at least  $9 - \varepsilon$  (including parts assigned to a bin but not packed in it) to ensure  $\text{BIGTHREAT} \leq m - \text{BIGBLOCK}$  continues to hold. We already note that for bins that are empty at the start of the fill-up phase the bound of  $9 - \varepsilon$  can be reached simply by using Next Fit (it will hold for all but at most one bin at any time).

**Rule 2** Whenever some item cannot be packed into some bin in  $\mathcal{D}$  that already received items of the same type in the fill-up phase (types are defined below), that bin will leave  $\mathcal{D}$  and  $\mathcal{U}$ . Each time we pack and/or assign  $10 + 6\varepsilon$  to  $\mathcal{D}$  in the fill-up phase, a new bin is added to  $\mathcal{D}$ . (Sometimes we will assign parts of items packed into other bins to bins in  $\mathcal{D}$ .)

**Rule 3** Each bin that is not in  $\mathcal{D}$  will receive at least  $10 + 6\varepsilon$  on average to maintain  $\text{TOPTHREAT} \leq u - d$ .

### Reducing the unused bins

We begin the fill-up phase by removing the bins in  $\mathcal{N}$  and  $\mathcal{Q}_5$  from the unused bins. The contents of these bins were and remain counted. For some later calculations it will still be important that these bins may exist, which is why we include them initially and gave a specific ordering for them.

Recall that the initial value of  $u$  is  $m - \ell - q_{\text{match}}$ . By the transition of the starting phase to the fill-up phase,  $\text{TOPTHREAT} \leq \frac{9-\varepsilon}{10+6\varepsilon}(m - c - 14)$ . So at least

$$\begin{aligned} m - \frac{9 - \varepsilon}{10 + 6\varepsilon}(m - c - 14) &= \frac{1 + 7\varepsilon}{10 + 6\varepsilon}m + \frac{9 - \varepsilon}{10 + 6\varepsilon} \cdot (14 + c) \\ &= \frac{1 + 7\varepsilon}{10 + 6\varepsilon}(m - c) + \frac{9 - \varepsilon}{10 + 6\varepsilon} \cdot 14 + c \end{aligned}$$

<sup>3</sup> The at most two bins in  $\Delta_{\text{nice},1} \cup \Delta_{\text{nice},2}$  can be placed anywhere. However, they are ignored when determining  $\beta_0$  later.



bins will not receive top items in the fill-up phase (but  $c$  of those bins are unavailable for top items anyway). Then after removing  $\mathcal{N}$  and  $\mathcal{Q}_5$  from the unused bins, we have  $u = m - c$ , so at least

$$\frac{1 + 7\varepsilon}{10 + 6\varepsilon} \cdot u + \frac{9 - \varepsilon}{10 + 6\varepsilon} \cdot 14$$

unused bins will not receive top items. We will initially set  $d = \frac{1+7\varepsilon}{10+6\varepsilon} \cdot u + \frac{9-\varepsilon}{10+6\varepsilon} \cdot 14$ .

Analogously,  $\text{BIGTHREAT} \leq m - c - 14$ , so at least  $c + 14$  bins will not receive big items, meaning that at least 14 unused bins will not receive big items in the fill-up phase. While packing the input, at any time there will be *half-full* bins. These are bins which have received some items during the fill-up phase but have not yet received (or been counted for)  $9 - \varepsilon$  or, in the case of non- $\mathcal{D}$  bins,  $10 + 6\varepsilon$ . These half-full bins need to be taken into account to ensure that Invariant 2.5 is maintained. Denote their number by  $h$ .

► **Invariant 4.2.** *At any time, the number of bins in  $\mathcal{D}$  that have not yet received any item in the fill-up phase is at least  $\frac{1+7\varepsilon}{10+6\varepsilon} \cdot u + \frac{9-\varepsilon}{10+6\varepsilon} \cdot 14 - h$ , where  $u = m - c$  initially and  $u$  is updated according to Rule 2.*

► **Lemma 4.3.** *As long as we pack items according to Rule 1 and update  $\mathcal{D}$  and  $\mathcal{U}$  according to Rule 2 for all except at most 10 bins, or pack items according to Rule 3, and at most 13 bins are half-full at any time,  $\text{TOPTHREAT} \leq u - d$ ,  $\text{BIGTHREAT} \leq m - \text{BIGBLOCK}$  and Invariant 4.2 are maintained.*

**Proof.** If we pack items according to Rule 3, the claims follow from the fact that we pack at least  $10 + 6\varepsilon$  in every non- $\mathcal{D}$  bin, decreasing  $\text{TOPTHREAT}$  and  $\text{BIGTHREAT}$  by at least 1 while increasing  $\text{BIGBLOCK}$  by at most 1 and removing exactly 1 bin from  $\mathcal{U}$  when we start using a new bin. Hence  $u - d$  and  $\text{TOPTHREAT}$  both decrease by 1, and  $\text{BIGTHREAT}$  decreases by at least 1. In this case the ratio  $d : u$  increases.

Regarding items that get packed according to Rule 1, we pack at least  $9 - \varepsilon$  in every bin in  $\mathcal{D}$  (and then remove such bins from  $\mathcal{D}$  and  $\mathcal{U}$ ) and add a new bin to  $\mathcal{D}$  after packing  $10 + 6\varepsilon$ , which means that we add a bin on average after using  $\frac{10+6\varepsilon}{9-\varepsilon}$  bins in  $\mathcal{D}$ . The ratio  $d : u$  remains constant during this process apart from at most one bin.

The ratio can be seen as follows. After packing a total size of  $x$  into  $\mathcal{D}$ , we have removed at most  $x/(9 - \varepsilon)$  bins from  $\mathcal{D}$  (because we only remove a bin from  $\mathcal{D}$  once we start using the next one) and we have added  $\lfloor x/(10 + 6\varepsilon) \rfloor$  bins to  $\mathcal{D}$ . Ignoring the rounding, overall  $d$  has decreased by at most  $x(\frac{1}{9-\varepsilon} - \frac{1}{10+6\varepsilon})$  and  $u$  has decreased by at most  $x/(9 - \varepsilon)$ . The ratio is maintained. The rounding means that the set  $\mathcal{D}$  may be 1 smaller during processing. This together with the initial value  $d = \frac{1+7\varepsilon}{10+6\varepsilon} \cdot u + \frac{9-\varepsilon}{10+6\varepsilon} \cdot 14$  leaves 10 bins for which the rules do not need to be followed, since  $\frac{9-\varepsilon}{10+6\varepsilon} \cdot 14 > 11$ .

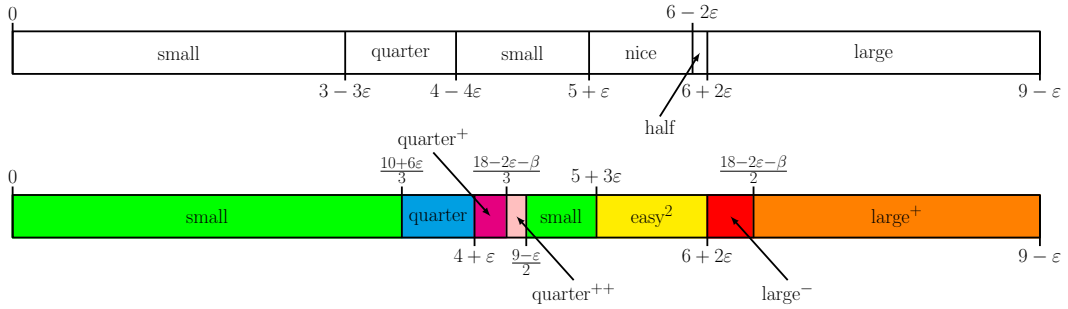
Finally, maintaining  $\text{BIGTHREAT} \leq m - \text{BIGBLOCK}$  given that initially  $\text{BIGTHREAT} \leq m - c - 14$  means that it is sufficient (due to the bin  $\Delta_{\text{nice},2}$ ) that at most 13 bins will be half-full at any time during the fill-up phase. ◀

A consequence of Invariant 4.2 is that the set  $\mathcal{D}$  does not become empty during the packing if we indeed maintain  $h \leq 13$ . Maintaining this invariant means that all remaining big and top items can be packed at any point during the fill-up phase. Note that if at some point indeed very many top items arrive, they can perhaps not all of them be packed outside of  $\mathcal{D}$ , as there can be various half-full bins outside of  $\mathcal{D}$ . However, by Invariant 4.2, as long as the total number of half-full bins is at most 13, all top items can indeed be packed.

**Item types**

Naturally, as we start filling up bins in the fill-up phase, new thresholds become important. Rather than leaving enough space for items that may arrive in the future as in the starting phase, we now want to use the remaining space efficiently. Bins in  $\mathcal{S}$  have at least  $(18 - 2\varepsilon) - (6 - 6\varepsilon) = 12 + 4\varepsilon$  space remaining and bins in  $\overline{\mathcal{Q}}_2$  leave at least  $(18 - 2\varepsilon) - 2(4 - 4\varepsilon) = 10 + 6\varepsilon$  space. If some item type fits at least three times on top of bins in  $\overline{\mathcal{Q}}_2$ , then First Fit gives a stronger bound on the packing and Rule 1 is satisfied. As there is at least  $10 + 6\varepsilon$  remaining space, we define small items to be of size at most  $\frac{10+6\varepsilon}{3}$ . Items of size more than  $\frac{9-\varepsilon}{2}$  that fit twice in this space are also small items. Apart from the range  $(5 + \varepsilon, 5 + 3\varepsilon]$ , these size ranges are a subset of what defined small items in the starting phase.

The next items are *quarter* items which may not fit three times on  $\overline{\mathcal{Q}}_2$  but at least three times on  $\mathcal{S}$ . For these we need to be slightly more careful; this is described below. *Small* items fit at least four times in an empty bin and at least two times on  $\mathcal{S} \cup \overline{\mathcal{Q}}_2$ . It can be seen that two items that are larger than small items satisfy Rule 1. To fill the remaining space well, the remaining items are split into *quarter*<sup>+</sup> and *quarter*<sup>++</sup> items. Analogously, *easy* items fit twice on  $\mathcal{S}$  and satisfy Rule 3, explaining the (unchanged) threshold  $6 + 2\varepsilon$ . These items have good sizes as well as large weights. To pack large items in the range  $(6 + 2\varepsilon, 9 - \varepsilon]$  efficiently we introduce a new threshold at  $\frac{18-2\varepsilon-\beta}{2}$  separating *large*<sup>-</sup> and *large*<sup>+</sup> items which will be explained later.



■ **Figure 4** (Sketch, using  $\varepsilon = 1/31, \beta = 5$ ) A comparison of the important thresholds for our algorithm in the starting phase (top) and in the fill-up phase (bottom). The small (green) items fit at least three times (resp. twice) in the remaining empty space in a  $\overline{\mathcal{Q}}_2$  bin ( $10 + 6\varepsilon$ ), the *easy*<sup>2</sup> (yellow) items fit at least twice in the remaining empty space in a  $\mathcal{S}$  bin ( $12 + 4\varepsilon$ ) and the *large*<sup>-</sup> (red) (resp. *quarter*<sup>+</sup> (purple)) items fit at least twice (resp. three times) in the remaining empty space in a bin filled to at most  $\beta$  ( $18 - 2\varepsilon - \beta$ ).

The value  $\beta$  will be defined later and will change during the fill-up phase; we will have  $\beta \in (3 - 3\varepsilon, 6 - 6\varepsilon]$ . There are ten size ranges, but the items in six ranges are straightforward to pack (see below). We call the *quarter*<sup>+</sup>, *quarter*<sup>++</sup>, *large*<sup>-</sup> and *large*<sup>+</sup> items *hard* items.

| Type                         | Max size                          | Type                      | Max size                          |
|------------------------------|-----------------------------------|---------------------------|-----------------------------------|
| small                        | $\frac{10+6\varepsilon}{3}$       | easy                      | $6 + 2\varepsilon$                |
| quarter                      | $4 + \varepsilon$                 | <i>large</i> <sup>-</sup> | $\frac{18-2\varepsilon-\beta}{2}$ |
| <i>quarter</i> <sup>+</sup>  | $\frac{18-2\varepsilon-\beta}{3}$ | <i>large</i> <sup>+</sup> | $9 - \varepsilon$                 |
| <i>quarter</i> <sup>++</sup> | $\frac{9-\varepsilon}{2}$         | big                       | $10 + 6\varepsilon$               |
| small                        | $5 + 3\varepsilon$                | top                       | 12                                |

We see that *quarter* items and small items may be slightly larger than in the starting phase. This does not decrease their weight. From the starting phase, we will only use the facts that bins in  $\overline{\mathcal{Q}}_2$  have level at most  $8 - 8\varepsilon$  and bins in  $\mathcal{S}$  have level at most  $6 - 6\varepsilon$ ; the old size thresholds play no further part in the analysis.

## 4.2 Packing methods for non-hard items

**small and big** These are items such that when packing them into  $\overline{Q}_2$  using First Fit (which are the fullest bins (ignoring  $\mathcal{X}$ ) that possibly still need to receive items in the fill-up phase), each bin (apart from constantly many) will receive at least  $9 - \varepsilon$  of items.

These items are always packed in the leftmost available bin that did not already receive items from another type in the fill-up phase. That is, we essentially use First Fit, but items of the three different size ranges are not packed together into bins. We only use bins in  $\mathcal{D}$ . This is feasible because new bins will enter the set  $\mathcal{D}$  as we pack items in it (Rule 2).

**easy and top** When packing easy or top items using First Fit into any bin that is not in  $\overline{Q}_2 \cup \Delta_{\text{large}} \cup \Delta_{\text{nice},1}$ , Rule 3 is automatically satisfied. (We have  $|\Delta_{\text{large}} \cup \Delta_{\text{nice},1}| \leq 2$ .) These items are packed exactly like the small items except that we skip the bins in  $\overline{Q}_2$ . We use First Fit for easy and top items (on the bins not in  $\overline{Q}_2$ ) separately.

**quarter** The quarter items are packed by First Fit, alternating between bins in  $\overline{Q}_2$  and bins not in  $\overline{Q}_2$ . If we run out of bins in  $\overline{Q}_2$  we use First Fit on dedicated bins not in  $\overline{Q}_2$ . If we run out of nonempty bins not in  $\overline{Q}_2$  we can always pack all remaining items as is shown in the full version.

**hard** These are the quarter<sup>+</sup>, quarter<sup>++</sup>, large<sup>-</sup> and large<sup>+</sup> items. These are the only types of items that we will sometimes (have to) pack into empty bins although nonempty bins that are not in  $\overline{Q}_2$  are still available. When packing hard items of one type using First Fit into empty bins, Rule 3 is satisfied *and* there is a surplus above the requirement of  $10 + 6\varepsilon$ . We use this surplus to pack as many items as possible into nonempty bins, which is necessary to succeed. See Section 4.3.

Once First Fit uses a new bin for an item, the bin previously considered (in which the item did not fit anymore) is removed from  $\mathcal{U}$ .

## 4.3 Hard (quarter<sup>+(+)</sup> and large) items

The above lemmas show that in the fill-up phase, all items except quarter<sup>+(+)</sup> and large items can be packed while following the rules – as long of course as quarter<sup>+(+)</sup> and large items are packed following the rules as well. We next consider quarter<sup>+(+)</sup> and large items separately. For this we first need to define some important sets.

### 4.3.1 Sets considered for packing hard items

#### The set $\mathcal{S}_{-L}$ and the parameter $\beta_0$

At the beginning of the fill-up phase, if there are  $e$  empty bins, then if we pack two large items in each empty bin, there will be  $e$  nonempty bins that will *not* be needed to pack large items that have not arrived yet, even if  $u$  of them arrive in total. In principle, we let  $\mathcal{S}_{-L}$  be the  $e$  *rightmost* nonempty bins. We only deviate from this if there are less than  $e$  nonempty bins. In that case  $\mathcal{S}_{-L}$  consists of all nonempty unused bins (after the removal of  $\mathcal{N} \cup Q_{2,5}$ ). We define  $\beta_0$  as the level of the leftmost bin in  $\mathcal{S}_{-L}$  unless all other nonempty bins are in  $\mathcal{N} \cup Q_{2,5}$ , in which case  $\beta_0$  is set to  $6 - 6\varepsilon$ . Of course, due to other items arriving, it may well be that some large items end up getting packed in  $\mathcal{S}_{-L}$  after all.

Since  $u = m - \ell - q_{\text{match}}$  at the start of the fill-up phase, it is not possible that more than  $u - q_5$  large items arrive in the fill-up phase. This holds because each bin counted in  $q_5 + q_{\text{match}}$  contains two items larger than  $6 - 2\varepsilon$  or a dominant item, and each bin counted in  $\ell$  contains two items larger than  $6 - 2\varepsilon$  or an item larger than 6. Moreover, as above, Invariant 4.2 is maintained by the  $(9 - \varepsilon)$ -guarantee. As soon as  $u$  starts to decrease in the fill-up phase, more than  $u - q_5$  large items may still arrive.

## 10:18 Improved Online Load Balancing with Known Makespan

### Construction of the set $\mathcal{D}$

We initially define  $\mathcal{D}$  as the leftmost  $\frac{1+7\varepsilon}{10+6\varepsilon} \cdot u + \frac{9-\varepsilon}{10+6\varepsilon} \cdot 14$  unused bins (where bins in  $\mathcal{Q}_5$  and  $\mathcal{N}$  have already been eliminated).

### Definition of the set $\mathcal{S}_L$

We define  $\mathcal{S}_L$  as the set of the remaining nonempty bins in  $\mathcal{U}$  (that is, all the unused bins that are not in  $\mathcal{D} \cup \mathcal{S}_{-L} \cup \mathcal{E}$ ). The set  $\mathcal{S}_L$  may be empty.

Note that the above construction is done only once. The sets  $\mathcal{S}_L$  and  $\mathcal{S}_{-L}$  do not increase and a bin leaves such a set only if the bin leaves  $\mathcal{U}$  or is added to  $\mathcal{D}$ . While packing items in the fill-up phase, bins will be added to  $\mathcal{D}$  from left to right. Hence entering  $\mathcal{D}$  will happen first to bins in  $\mathcal{S}_L$  and then (possibly) to  $\mathcal{S}_{-L}$  and  $\mathcal{E}$ . Such bins leave their original sets.

Bins in  $\mathcal{S}$  are filled to at most  $6 - 6\varepsilon$  and hence have at least  $12 + 4\varepsilon$  empty space. Easy<sup>2</sup> items fit at least twice. Bins in  $\mathcal{S}_{-L}$  are filled to at most  $\beta_0$  and hence have at least  $18 - 2\varepsilon - \beta_0$  empty space. Large<sup>-</sup> items therefore fit at least twice (explaining that threshold). Bins initially in  $\mathcal{D}$  are loaded to at most  $2(4 - 4\varepsilon)$  and hence have at least  $18 - 2\varepsilon - 2(4 - 4\varepsilon) = 10 + 6\varepsilon$  space.

### 4.3.2 Packing methods for hard items: Five stages

Hard items are packed as follows. There are five possible stages; depending on what the packing looks like when the fill-up phase starts, not all stages might be applicable. Generally, we start by taking advantage of any  $\mathcal{Q}_{1,5}$  bins that are available, then we start filling the nonempty bins from right to left, always trying to avoid using empty bins as much as possible.

Hard items are always distributed among several types of bins ( $\mathcal{D}, \mathcal{S}_L, \mathcal{S}_{-L}, \mathcal{E}$ ). One of the used types will always be  $\mathcal{D}$ . In several cases, we will specify that nonintegral numbers of bins need to be used for some items. This can be implemented as follows. We always start by using a bin that is not in  $\mathcal{D}$ . Then, we keep using bins in  $\mathcal{D}$  until we would exceed the desired ratio. At that point we use a bin that is not in  $\mathcal{D}$  again, and repeat the process. In the long term we get closer and closer to the desired ratio.

#### Stage 1

In this stage, as mentioned above we first exploit any bins in  $\mathcal{Q}_{1,5}$  that may exist. These bins can be intermixed with  $\overline{\mathcal{Q}}_1$  in the sorted order, but we use them first. To be more precise, these bins are not used to pack any new items (as they are already quite full) but rather to pack additional items into  $\mathcal{D}$ , as follows. Note that these bins are not in  $\mathcal{U}$ . For this stage we define the upper bound for quarter<sup>+</sup> items as  $\frac{9-\varepsilon}{2}$  and the upper bound for large<sup>-</sup> items as  $9 - \varepsilon$  and hence neither quarter<sup>++</sup> items nor large<sup>+</sup> items exist. This also means that the value of  $\beta$  (see table of item type thresholds for the fill-up phase) does not yet play a role.

**Quarter<sup>+(+)</sup>:** As long as there exist bins in  $\mathcal{Q}_{1,5}$  with partially uncounted contents, for packing quarter<sup>+(+)</sup> items such bins are considered to contain quarter<sup>+(+)</sup> items (of the size of the ignored items). Each such bin allows us to pack  $\frac{1+7\varepsilon}{1-3\varepsilon}$  bins in  $\mathcal{D}$  with two quarter<sup>+(+)</sup> items each. In these  $\frac{1+7\varepsilon}{1-3\varepsilon}$  bins we then pack (or count) more than  $2\frac{1+7\varepsilon}{1-3\varepsilon}(4 + \varepsilon) + (1 + 7\varepsilon) = \frac{1+7\varepsilon}{1-3\varepsilon} \cdot (9 - \varepsilon)$ , which includes the so far uncounted part of the bin in  $\mathcal{Q}_{1,5}$ .

**Large:** As long as there exist bins in  $\mathcal{Q}_{1,5}$  with partially uncounted contents, for packing large items such bins are considered to contain large items (of the size of the ignored items). Each such bin allows us to pack  $\frac{1+7\varepsilon}{3-3\varepsilon}$  bins in  $\mathcal{D}$  with one large item each. In these  $\frac{1+7\varepsilon}{3-3\varepsilon}$  bins we then pack (or count) more than  $\frac{1+7\varepsilon}{3-3\varepsilon}(6+2\varepsilon) + (1+7\varepsilon) = \frac{1+7\varepsilon}{3-3\varepsilon} \cdot (9-\varepsilon)$ , which includes the so far uncounted part of the bin in  $\mathcal{Q}_{1,5}$ .

It can be seen that by packing  $\text{quarter}^{+(+)}$  and large items this way, Rule 1 and Rule 3 are followed. Once we run out of bins in  $\mathcal{Q}_{1,5}$ , we start using the methods in the following table. The value of  $\beta$  will change over time and is set at the start of each of the following stages. Note here that for  $\beta \leq \frac{9-\varepsilon}{2}$  the upper bound for  $\text{quarter}^+$  items is at least the upper bound for  $\text{quarter}^{++}$  items and hence  $\text{quarter}^{++}$  items do not exist as long as  $\beta \leq \frac{9-\varepsilon}{2}$ .

| Item type             | Bin type   | Nr bins   | Per bin | Counted                              | Average                      |
|-----------------------|--|---|---------|--------------------------------------|------------------------------|
| $\text{quarter}^+$    | $\mathcal{D}$  | 3   | 2       | $2(4+\varepsilon)$                   | $9 + \frac{9}{4}\varepsilon$ |
|                       | $\overline{\mathcal{Q}}_1, \mathcal{S}_{-L}(\mathcal{S}_L), \mathcal{E}$ | 1   | 3       | $3(4+\varepsilon)$                   |                              |
| $\text{quarter}^{++}$ | $\mathcal{E}$  | 1   | 4       | $\frac{4}{3}(18-2\varepsilon-\beta)$ | $9-\varepsilon$              |
|                       | $\mathcal{D}$  | $\frac{45-4\beta-5\varepsilon}{2\beta-(9-\varepsilon)}$ | 2       | $\frac{2}{3}(18-2\varepsilon-\beta)$ |                              |
| $\text{large}^-$      | $\mathcal{D}$  | 1   | 1       | $6+2\varepsilon$                     | $9+3\varepsilon$             |
|                       | $\overline{\mathcal{Q}}_1, \mathcal{S}_{-L}(\mathcal{S}_L), \mathcal{E}$ | 1   | 2       | $12+4\varepsilon$                    |                              |
| $\text{large}^+$      | $\mathcal{D}$  | $\frac{18-2\varepsilon}{\beta} - 2$                     | 1       | $\frac{18-2\varepsilon-\beta}{2}$    | $9-\varepsilon$              |
|                       | $\mathcal{E}$  | 1   | 2       | $18-2\varepsilon-\beta$              |                              |

The third column indicates the number of bins used each time. The column “Average” contains the average amount counted over all bin types used for this item.

This table is implemented as follows. Items from these four types are packed into separate bins. For each type except  $\text{quarter}^{++}$  items, we first use a bin in  $\mathcal{D}$ . For  $\text{quarter}^{++}$  items, we use  $\mathcal{E}$  first to ensure that the average packed in the bins with these items is always greater than  $9-\varepsilon$ , apart from at most one bin: the bin currently being used. For all other types, we have this guarantee for all used bins up to and including the most recently filled bin in  $\mathcal{E}$ , which is all but at most four bins for each type.

For each bin used we pack items in it until we have packed the number of items in the column Per bin. We first use bins in  $\mathcal{D}$  until we have packed the number of items in the column Per bin and until we have used the number of bins in the column Nr bins, followed by one bin in  $\mathcal{E}$ . (If the number of bins supposed to be used in  $\mathcal{D}$  is not an integer, then the number of bins used in  $\mathcal{D}$  is always off by less than 1 compared to the desired ratio of  $\mathcal{D} : \mathcal{E}$  usage.) This procedure keeps repeating. Whenever we start using a new bin, the previous bin for this type is called *closed*. Once a bin is closed it is removed from  $\mathcal{U}$ .

In the following we will always check if bins with level at most  $\beta$  exist. While this is true we remain in the current stage. Else we update  $\beta$  to the next higher bound.

## Stage 2

We set  $\beta := \min(\beta_0, 4-4\varepsilon)$ . As long as there are bins with level at most  $\beta$  available, we use these bins (including bins in  $\mathcal{S}_L$  if needed) for packing  $\text{quarter}^+$  and  $\text{large}^-$  items. There are no  $\text{quarter}^{++}$  items and only  $\text{large}^+$  items are packed into empty bins.

## Stage 3

As Stage 2, but with  $\beta := \min(\beta_0, \frac{9-\varepsilon}{2})$ . Bins in  $\mathcal{S}_L$  with level at most  $\beta$  will still be used for  $\text{quarter}^+$  and  $\text{large}^-$  items. There are no  $\text{quarter}^{++}$  items and only  $\text{large}^+$  items are packed into empty bins.

**Stage 4**

This stage only exists if  $\beta_0 > \frac{9-\varepsilon}{2}$ . We set  $\beta := \beta_0$ .

Quarter<sup>++</sup> and large<sup>+</sup> items are packed into empty bins. All items are packed according to the table. Bins in  $\mathcal{S}_L$  are *not* used for quarter<sup>+</sup> items; we would go to Stage 5 instead. This is done to ensure that enough bins are left for large<sup>-</sup> items and that we do not waste bins on quarter<sup>+</sup> items.

**Stage 5**

Only bins in  $\mathcal{S}_L \cup \mathcal{D} \cup \mathcal{E}$  are left. At this point quarter<sup>+</sup> and large<sup>-</sup> items are also packed into empty bins.

Stage 6 would be the stage where all empty bins have been filled while some bins in  $\mathcal{S}_L \setminus \mathcal{D}$  remain. The case where all nonempty bins get filled first is discussed in the full version. We will show that Stage 6 is not reached or we are in a good situation.

As is in the starting phase, if some item cannot be packed according to the packing rules (including the case where we change the packing rules if we reach a good situation) we use the rule of last resort.

---

**References**


---



- 1 Islam Akaria and Leah Epstein. Bin stretching with migration on two hierarchical machines. *Math. Methods Oper. Res.*, 98(1):111–153, 2023.
- 2 Susanne Albers and Matthias Hellwig. Semi-online scheduling revisited. *Theoretical Computer Science*, 443:1–9, 2012.
- 3 Yossi Azar and Oded Regev. On-line bin-stretching. In *RANDOM*, volume 1518 of *Lecture Notes in Computer Science*, pages 71–81. Springer, 1998.
- 4 Yossi Azar and Oded Regev. On-line bin-stretching. *Theoretical Computer Science*, 268(1):17–41, 2001.
- 5 János Balogh, József Békési, György Dósa, Jiří Sgall, and Rob van Stee. The optimal absolute ratio for online bin packing. *J. Comput. Syst. Sci.*, 102:1–17, 2019. doi:10.1016/j.jcss.2018.11.005.
- 6 Martin Böhm, Matej Lieskovský, Sören Schmitt, Jiří Sgall, and Rob van Stee. Improved online load balancing with known makespan. *arXiv e-prints*, page arXiv:2407.08376, 2024. doi:10.48550/arXiv.2407.08376.
- 7 Martin Böhm, Jiří Sgall, Rob van Stee, and Pavel Veselý. Online bin stretching with three bins. *Journal of Scheduling*, 20(6):601–621, 2017.
- 8 Martin Böhm, Jiří Sgall, Rob Van Stee, and Pavel Veselý. A two-phase algorithm for bin stretching with stretching factor 1.5. *Journal of Combinatorial Optimization*, 34(3):810–828, 2017.
- 9 Martin Böhm and Bertrand Simon. Discovering and certifying lower bounds for the online bin stretching problem. *Theor. Comput. Sci.*, 938:1–15, 2022.
- 10 Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *SIGACT News*, 47(3):93–129, August 2016. doi:10.1145/2993749.2993766.
- 11 Jérôme Dohrau. Online makespan scheduling with sublinear advice. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 177–188. Springer, 2015.
- 12 Leah Epstein. Bin stretching revisited. *Acta Informatica*, 39(2):97–117, 2003.
- 13 Rudolf Fleischer and Michaela Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000.
- 14 Michaël Gabay, Nadia Brauner, and Vladimir Kotov. Improved lower bounds for the online bin stretching problem. *4OR*, 15(2):183–199, 2017.

- 15 Michaël Gabay, Vladimir Kotov, and Nadia Brauner. Online bin stretching with bunch techniques. *Theoretical Computer Science*, 602:103–113, 2015.
- 16 Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell system technical journal*, 45(9):1563–1581, 1966.
- 17 Hans Kellerer and Vladimir Kotov. An efficient algorithm for bin stretching. *Operations Research Letters*, 41(4):343–346, 2013.
- 18 Hans Kellerer, Vladimir Kotov, and Michaël Gabay. An efficient algorithm for semi-online multiprocessor scheduling with given total processing time. *J. Sched.*, 18(6):623–630, 2015. doi:10.1007/s10951-015-0430-4.
- 19 Hans Kellerer, Vladimir Kotov, Maria Grazia Speranza, and Zsolt Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.
- 20 Antoine Lhomme, Nicolas Catusse, and Nadia Brauner. Computational bounds on randomized algorithms for online bin stretching. *CoRR*, abs/2405.19071, 2024. doi:10.48550/arXiv.2405.19071.
- 21 Antoine Lhomme, Olivier Romane, Nicolas Catusse, and Nadia Brauner. Online bin stretching lower bounds: Improved search of computational proofs. *CoRR*, abs/2207.04931, 2022.
- 22 Matej Lieskovský. Better algorithms for online bin stretching via computer search. *CoRR*, abs/2201.12393, 2022. arXiv:2201.12393.
- 23 Matej Lieskovský. Better algorithms for online bin stretching via computer search. In *LATIN (1)*, volume 14578 of *Lecture Notes in Computer Science*, pages 241–253. Springer, 2024.
- 24 Marc P Renault, Adi Rosén, and Rob van Stee. Online algorithms with advice for bin packing and scheduling problems. *Theoretical Computer Science*, 600:155–170, 2015.
- 25 John F Rudin III. *Improved bounds for the online scheduling problem*. PhD thesis, The University of Texas at Dallas, 2001.





# On the NP-Hardness Approximation Curve for Max-2Lin(2)

Björn Martinsson  

KTH Royal Institute of Technology, Stockholm, Sweden

---

## Abstract

In the Max-2Lin(2) problem you are given a system of equations on the form  $x_i + x_j \equiv b \pmod{2}$ , and your objective is to find an assignment that satisfies as many equations as possible. Let  $c \in [0.5, 1]$  denote the maximum fraction of satisfiable equations. In this paper we construct a curve  $s(c)$  such that it is NP-hard to find a solution satisfying at least a fraction  $s$  of equations. This curve either matches or improves all of the previously known inapproximability NP-hardness results for Max-2Lin(2). In particular, we show that if  $c \geq 0.9232$  then  $\frac{1-s(c)}{1-c} > 1.48969$ , which improves the NP-hardness inapproximability constant for the min deletion version of Max-2Lin(2). Our work complements the work of O'Donnell and Wu that studied the same question assuming the Unique Games Conjecture.

Similar to earlier inapproximability results for Max-2Lin(2), we use a gadget reduction from the  $(2^k - 1)$ -ary Hadamard predicate. Previous works used  $k$  ranging from 2 to 4. Our main result is a procedure for taking a gadget for some fixed  $k$ , and use it as a building block to construct better and better gadgets as  $k$  tends to infinity. Our method can be used to boost the result of both smaller gadgets created by hand ( $k = 3$ ) or larger gadgets constructed using a computer ( $k = 4$ ).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Constraint and logic programming; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Inapproximability, NP-hardness, 2Lin(2), Max-Cut, Gadget

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.11

**Category** APPROX

**Supplementary Material**

*Software (Source code):* <https://github.com/bjorn-martinsson/NP-hardness-of-Max-2Lin-2>  
archived at `swh:1:dir:24803e393eb5826edd7af8e837b182cb24282691`

## 1 Introduction

Maximum constraint satisfaction problems (Max-CSPs) form one of the most fundamental classes of problems studied in computational complexity theory. A Max-CSP is a type of problem where you are given a list of variables and a list of constraints, and your goal is to find an assignment that satisfies as many of the constraints as possible. Some common examples of Max-CSP are Max-Cut and Max-2Sat. Every Max-CSP also has a corresponding Min-CSP-deletion problem where your objective is deleting as few constraints as possible to make all of the remaining constraints satisfiable. The Min-CSP-deletion problem is fundamentally the same optimisation problem as its corresponding Max-CSP, however their objective values are different.

### 1.1 History of Max-Cut

The Max-Cut problem is arguably both the simplest Max-CSP as well as the simplest NP-hard problem. In the Max-Cut problem you are given an undirected graph, and your objective is to find a cut of the largest possible size. A cut of an undirected graph is a



© Björn Martinsson;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 11; pp. 11:1–11:38



Leibniz International Proceedings in Informatics

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

partition of the vertices into two sets and the size of a cut is the fraction of edges that connect the two sets relative to the total number of edges. Solving Max-Cut exactly is difficult, but there are trivial approximation algorithms that get within a factor of  $\frac{1}{2}$  of the optimum. One such algorithm is randomly picking the cut by tossing one coin per vertex.

Knowing this, one natural question is, how close can a polynomial time algorithm get to the optimum? Goemans and Williamson partly answered this in a huge breakthrough in 1995 [8] by applying semi-definite programming (SDP) to create a polynomial time algorithm that finds a solution that is within a factor of  $\alpha_{\text{GW}} \approx 0.87856$  of the optimum. At the time, there was hope that Goemans and Williamson’s algorithm could be improved further to get even better approximation factors than 0.87856, but no such improvements were ever found. Instead, in 2004 Khot et al [11] proved using the Unique Games Conjecture (UGC), that approximating Max-Cut within a factor of  $\alpha_{\text{GW}} + \varepsilon$  is NP-hard for any  $\varepsilon > 0$ . This conjecture had been introduced by Khot two years prior [12]. This was possibly the first result establishing the close connection between UGC and SDP based algorithms.

To this day, UGC remains an open problem, and in particular no one has been able to find an approximation algorithm for Max-Cut with a better approximation ratio than  $\alpha_{\text{GW}}$ . In 2008 O’Donnell and Wu [14] were able to very precisely describe the tight connection between SDP based approximation algorithms for Max-Cut and UGC. They constructed a curve  $\text{Gap}_{\text{SDP}}(c) : [0.5, 1] \rightarrow [0.5, 1]$  with the following two properties:

1. It is UGC-hard to find a cut of size  $\text{Gap}_{\text{SDP}}(c) + \varepsilon$  given that the optimal cut has size  $c$  for any  $\varepsilon > 0$ . We here use *UGC-hard* as a short hand for “NP-hard under UGC”.
2. Within the RPR<sup>2</sup>-framework [7, 14], there are polynomial time algorithms that are guaranteed to find a cut of size at least  $\text{Gap}_{\text{SDP}}(c) - \varepsilon$  if the optimal cut has size  $c$ . The RPR<sup>2</sup>-framework is a generalisation of Goemans and Williamson’s algorithm.

This means that their work both describe the best known polynomial time approximation algorithms for Max-Cut, and also show that under UGC these approximation algorithms cannot be improved. It is important to note that their algorithmic results do not require UGC. We emphasize that one implication of their result is that giving efficient algorithms with a better performance would disprove UGC.

## 1.2 NP-hardness inapproximability of Max-2Lin(2)

Max-2Lin(2) is a Max-CSP that is very closely related to Max-Cut. An instance of Max-2Lin(2) is a system of linear equations on the form  $x_i + x_j \equiv b \pmod{2}$ , and the objective is to find an assignment that satisfies as many equations as possible. Max-Cut is the special case where we only allow equations with right hand side equal to 1. This implies that any hardness result for Max-Cut immediately yields the same hardness result for Max-2Lin(2). One example of this is the UGC-hardness of Max-Cut described by the  $\text{Gap}_{\text{SDP}}(c)$  curve by O’Donnell and Wu [14].

Furthermore, O’Donnell and Wu’s algorithmic results [14] also directly carries over to Max-2Lin(2). This is because the RPR<sup>2</sup>-framework that they relied on uses odd rounding functions, and therefore does not differentiate between Max-Cut and Max-2Lin(2).

The conclusion is that the  $\text{Gap}_{\text{SDP}}(c)$  describes a tight connection between the UGC-hardness of Max-2Lin(2) as well as the best known polynomial time approximation algorithms for Max-2Lin(2). On the other hand, the NP-hardness inapproximability of Max-2Lin(2) is not well understood. The strongest NP-hardness inapproximability results known for Max-2Lin(2) ([9], [16]) are still far off from the UGC-hardness described by the  $\text{Gap}_{\text{SDP}}(c)$  curve.

The aim of this paper is to improve the state of the art NP-hardness inapproximability of Max-2Lin(2) and also to give the full picture of the state of the art NP-hardness inapproximability of Max-2Lin(2). We do this by constructing a curve  $s(c) : [0.5, 1] \rightarrow [0.5, 1]$  such that it is NP-hard to distinguish between instances where the optimal assignment satisfies a fraction of  $c$  of the equations, and instances where all assignments satisfy at most a fraction of  $s(c)$  of the equations. Our curve either matches or improves all previously known NP-hardness inapproximability results for Max-2Lin(2). We construct the curve by solving a separate optimisation problem for each value of  $c$ , so our result covers the entire spectrum of  $c \in [0.5, 1]$ .

Our result complements the work by O’Donnell and Wu [14]. Our curve describes the state of the art NP-hardness inapproximability of Max-2Lin(2) while O’Donnell and Wu’s  $\text{Gap}_{\text{SDP}}(c)$  curve describes the UGC-hardness of Max-2Lin(2). It is worth noting that UGC is still an open problem that over the years has been the subject of much debate. There are results that indicate that UGC might be true, such as the proof of the closely related 2-to-2 Games Conjecture [13]. But on the other hand there are also results that indicate the UGC might be false, such as the existence of subexponential algorithms for Unique Games [2]. Currently there is no consensus for whether UGC is true or not. It is for this reason that it is important to study NP-hardness independent of UGC, especially for fundamental problems such as Max-2Lin(2).

### 1.3 Gadget reductions

Gadgets are the main tools used to create reductions from one Max-CSP  $\Phi$  to another Max-CSP  $\Psi$ . A gadget is a description of how to translate a specific constraint  $\varphi$  of  $\Phi$  into one or more constraints of  $\Psi$ . For example, if  $\Phi$  is Max-3Lin(2) and  $\Psi$  is Max-Cut, then a gadget from  $\varphi$  to  $\Psi$  is a graph. A gadget is allowed to use both the original variables in the constraint  $\varphi$ , which are called *primary variables*, and new variables specific to the gadget, which are called *auxiliary variables*.

The standard technique used to construct gadgets is to follow the “automated gadget” framework of Trevisan et al [15]. This framework describes how to construct a gadget by solving a linear program and also proves that the constructed gadget is optimal. This framework is mainly used to construct gadgets for small and simple Max-CSPs. This is because the number of variables in the gadget scales exponentially with the number of satisfying assignments of  $\varphi$ . Furthermore, the number of constraints in the LP scales exponentially with the number of variables, so it scales double exponentially with the number of satisfying assignment of  $\varphi$ .

As an example let us take the gadget from Max-3Lin(2) to Max-2Lin(2) used by Håstad [9], which was originally constructed by Trevisan et al [15]. A constraint in Max-3Lin(2) has 4 satisfiable assignments. Having 4 satisfiable assignments means that the gadget uses  $2^4 = 16$  variables. Furthermore, since Max-2Lin(2) allow negations, half of these variables can be removed because of negations. So the actual number of variables in the gadget is  $2^{4-1} = 8$ . This in turn implies that the number of constraints in the LP is  $2^8 = 256$ . This number is small enough that it is feasible for a computer to solve the LP. In this paper we are interested in constructing gadgets from generalisations of Max-3Lin(2), called the Hadamard Max-CSPs. These have significantly more satisfying assignments than Max-3Lin(2). It is easy to see that a simple-minded application of the “automated gadget” framework leads to an LP that is far too large to naively be solved by a computer. This means that we have to deviate from the “automated gadget” framework in order to construct our gadgets.

Gadgets have two important properties, called *soundness*  $s$  and *completeness*  $c$ . If a gadget is constructed using the “automated gadget” framework, then it is trivial to calculate the completeness of the gadget. On the other hand, calculating the soundness of a gadget from  $\Phi$  to  $\Psi$  involves solving instances of  $\Psi$ . In practice, calculating the soundness of a large gadget can be very difficult since  $\Psi$  is usually an NP-hard problem.

Gadgets can be constructed with different goals in mind. The case that we are interested in is finding the gadget with the largest soundness for a fixed completeness. This is what allows us to construct our curve  $s(c)$ . In general there are also other objectives that could be of interest when constructing gadgets. One such case is finding the gadget with the smallest ratio of  $\frac{s}{c}$ . This corresponds to finding the best lower bound for the approximation ratio of Max-2Lin(2). Another possibility is to maximise  $\frac{1-s}{1-c}$ . This corresponds to finding the best upper bound for the approximation ratio of Min-2Lin(2)-deletion. It is possible to use the “automated gadget” framework by Trevisan et al [15] to find the optimal gadgets for all of these scenarios.

## 1.4 The Hadamard Max-CSPs Max-Had $_k$

One of the earliest gadget reductions used to show NP-hardness inapproximability of Max-2Lin(2) is a gadget reduction from Max-3Lin(2) used by Håstad in his classical paper from 1997 [9], which was constructed by Trevisan et al [15]. More recently, NP-hardness inapproximability results for Max-2Lin(2) have used gadget reductions from a generalisation of Max-3Lin(2) called the Hadamard Max-CSPs [10, 16]. The  $(2^k - 1)$ -ary Hadamard Max-CSP,  $k \geq 2$ , is a constraint satisfaction problem where a clause is satisfied if and only if its literals form the truth table of a linear  $k$ -bit Boolean function. The  $(2^k - 1)$ -ary Hadamard CSP is denoted by Max-Had $_k$ . One special case is  $k = 2$ , where the number of literals of a clause is 3. It turns out that this case coincides with Max-3Lin(2). This means that Max-Had $_k$  can be seen as a generalisation of Max-3Lin(2).

There are mainly two reasons as to why Max-Had $_k$  is useful for gadget reductions. The first reason is that Max-Had $_k$  is a very sparse CSP. It being sparse refers to the number of satisfiable assignments of a clause being few in relation to the total number of possible assignments. The number of satisfying assignments of a clause is just  $2^k$ , one for each linear  $k$ -bit Boolean function, while the total number of possible assignments is  $2^{(2^k - 1)}$ .

The second reason is that Max-Had $_k$  is a *useless predicate* for any  $k \geq 2$ , which is an even stronger property than being approximation resistant. This was shown by Chan in 2013 [6]. Max-Had $_k$  being a useless predicate means that if you are given a nearly satisfiable instance of Max-Had $_k$ , then it is NP-hard to find an assignment such that the distribution over the  $(2^k - 1)$  long bit strings given by the literals of the clauses is discernibly different from the uniform distribution.

### 1.4.1 Historical overview of Had $_k$ -to-2Lin(2) gadgets

In 1996, Trevisan et al [15] constructed the optimal gadget from Max-Had $_2$  to Max-2Lin(2). They showed that the Max-Had $_2$  gadget that minimises  $\frac{s}{c}$  is the same gadget as the one that maximises  $\frac{1-s}{1-c}$ . Furthermore, since this gadget is very small, using only 8 variables, they were able to construct it using the “automated gadget” framework.

In 2015, Håstad et al. [10] constructed gadgets from Max-Had $_3$  to Max-2Lin(2). They showed that the Max-Had $_3$  gadget that minimises  $\frac{s}{c}$  is equivalent to the Max-Had $_2$  gadget. So using Max-Had $_3$  over Max-Had $_2$  does not give an improved hardness for the approximation ratio of Max-2Lin(2). However, the Max-Had $_3$  gadget that maximises  $\frac{1-s}{1-c}$  is notably better

than the Max-Had<sub>2</sub> gadget. This gadget is relatively small, only using 128 variables. This is too many variables for it to be possible to naively apply the “automated gadget” framework. However, Håstad et al. were still able to construct and analyse the optimal gadget by hand based on ideas from the “automated gadget” framework.

In 2018, Wiman [16] constructed gadgets from Max-Had<sub>4</sub> to Max-2Lin(2). Note that Max-Had<sub>4</sub> gadgets have  $2^{15}$  variables. Calculating the soundness of a such a gadget requires solving an instance of Max-2Lin(2) with  $2^{15}$  variables, which is infeasible to do by hand or even with a computer. Wiman initially followed the “automated gadget” framework. However, in order to be able to calculate the soundness of the gadget, Wiman relaxed the Max-2Lin(2) problem into a Max-Flow problem. This *relaxed soundness*  $rs$  is an upper bound of the true soundness. This relaxation made it possible for Wiman to use a computer to find the gadget that maximises  $\frac{1-rs}{1-c}$ . Wiman’s relaxation was successful, since by using it he was able to find a Max-Had<sub>4</sub> gadget that was better than the optimal Max-Had<sub>3</sub> gadget. Note, however, that by using a relaxation, it is uncertain whether Wiman found the optimal Max-Had<sub>4</sub> gadget or not.

### 1.4.2 Our Had<sub>k</sub>-to-2Lin(2) gadgets

In this paper, we construct gadgets from Max-Had<sub>k</sub> to Max-2Lin(2) for  $k$  approaching infinity. Recall that a gadget uses  $2^{2^k-1}$  variables, so using a computer to construct gadgets for  $k > 5$  is normally impossible. We get around this limitation by introducing a procedure for taking Max-Had<sub>k</sub> gadgets and transforming them to Max-Had<sub>k'</sub> gadgets, for  $k' > k$ . We refer to this procedure as the *lifting* of a Max-Had<sub>k</sub> gadget into a Max-Had<sub>k'</sub> gadget. Two of the properties of lifting is that the completeness stays the same and the soundness does not decrease.

To show NP-hardness of approximating Max-2Lin(2), we start by constructing Max-Had<sub>k</sub> to Max-2Lin(2) gadgets for  $k = 4$  using a computer. We then analytically prove an upper bound of Wiman’s relaxed soundness of the lifting of these gadgets as  $k' \rightarrow \infty$ .

The method we use to construct our gadgets is by solving an LP. This LP is similar to what Wiman could have used to construct his gadget. The difference is that the LP we use is made to minimise the soundness of the lifted gadget, instead of minimising the soundness of the gadget itself. If done naively, this LP would have roughly  $2^{3 \cdot (2^k-1)} = 2^{45}$  variables. But by making heavy use of symmetries of the LP, we are able to bring it down to a feasible size.

The main technical work of this paper is proving an upper bound on Wiman’s relaxed soundness of a lifted gadget as  $k' \rightarrow \infty$ . Recall that calculating Wiman’s relaxed soundness involves solving instances of Max-Flow. As  $k'$  tends to infinity, the size of these instances also tend to infinity. In order to lower bound the value of these Max-Flow problems, we introduce the concept of a type of infeasible flows which we call *leaky flows*. A leaky flow is a flow for which the conservation of flows constraint has been relaxed. This allows leaky flows to attain higher values compared to feasible flows. We then show that by randomly overlapping leaky flows onto the large Max-Flow instances, we are able to get closer and closer to a feasible flow as the size of the instances tend to infinity.

## 1.5 Our results and comparison to previous results

Using a gadget reductions from Max-Had<sub>k</sub> to Max-2Lin(2), we are able to construct a curve  $s(c) : [0.5, 1] \rightarrow [0.5, 1]$  such that it is NP-hard to distinguish between instances of Max-2Lin(2) where the optimal assignment satisfies a fraction of  $c$  of the equations and instances where all assignments satisfy at most a fraction of  $s(c)$  of the equations. This curve does not have an explicit formula. Instead, each point on the curve is defined as the solution to an LP, which we solve using a computer.

## 11:6 On the NP-Hardness Approximation Curve for Max-2Lin(2)

► **Theorem 1.1.** *Let  $s(c) : [0.5, 1] \rightarrow [0.5, 1]$  be the curve defined in Definition 4.1. Then for every sufficiently small  $\varepsilon > 0$ , it is NP-hard to distinguish between instances of Max-2Lin(2) such that*

**Completeness** *There exists an assignment that satisfies a fraction at least  $c - \varepsilon$  of the constraints.*

**Soundness** *All assignments satisfy at most a fraction  $s(c) + \varepsilon$  of the constraints.*

A notable point on the curve is  $c = \frac{590174949}{639271832} \approx 0.9232$  and  $s(c) = \frac{141533171}{159817958} \approx 0.8856$ . This is the point on the curve that gives the highest NP-hardness inapproximability factor  $\frac{1-s}{1-c}$  of Min-2Lin(2)-deletion.

► **Corollary 1.2.** *It is NP-hard to approximate Min-2Lin(2)-deletion within a factor of  $\frac{73139148}{49096883} + \varepsilon \approx 1.48969 + \varepsilon$ .*

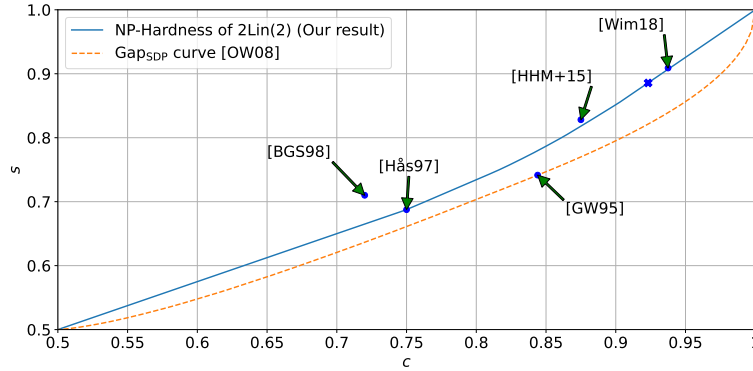
In order to be able to compare our curve  $s(c)$  to prior results, we plot our curve together with O’Donnell and Wu’s  $\text{Gap}_{\text{SDP}}(c)$  curve [14], which, as discussed earlier, describes both the UGC-hardness of Max-2Lin(2), as well as the best known polynomial time approximation algorithms of Max-2Lin(2). Additionally, we also include historical NP-hardness inapproximability results as points in the diagram. We have also marked the point  $(c, s)$  where Goemans and Williamson’s algorithm achieves the approximation ratio of  $\frac{s}{c} = \alpha_{\text{GW}} \approx 0.87856$ . This point was shown to be UGC-hard by Khot et al. in 2004 [11].

The curve  $s(c)$  is plotted in three Figures. All three Figures contain the same exact same data, but the data is plotted in different ways. In Figure 1 the soundness  $s(c)$  is on the  $y$ -axis and the completeness  $c$  is on the  $x$ -axis. This plot has the disadvantage that to the eye, it is difficult to distinguish the exact shape of the curve  $s(c)$ . In the next plot, Figure 2,  $\frac{s(c)}{c}$  is on the  $y$ -axis and  $c$  is on the  $x$ -axis. This plot describes the approximation ratio of Max-2Lin(2). The third plot, in Figure 3, has  $\frac{1-s(c)}{1-c}$  on the  $y$  axis and  $c$  on the  $x$ -axis. This plot describes the approximation ratio of Min-2Lin(2)-deletion.

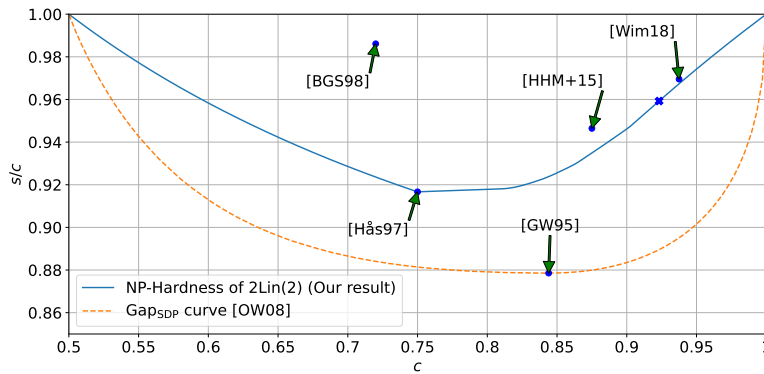
It is important to note that the curves in Figure 1 are convex functions since it is possible to take the convex combination of two hard instances using disjoint sets of variables. One implication from this is that it is possible to construct NP-hardness curves using any of the points  $(c, s)$  by drawing two lines, one from  $(0.5, 0.5)$  to  $(c, s)$  and one from  $(c, s)$  to  $(1, 1)$ . This means that all of the historical inapproximability results can also be described using convex curves.

In Figures 1-3 prior inapproximability results for Max-2Lin(2) are marked as dots. Bellare et al [5] was first to give an explicit NP-hardness result, which had  $c = 0.72$  and  $s = 0.71$ . In 2015, Håstad et al [10] used Chan’s result [6] to create a gadget reduction from Max-Had<sub>3</sub> which had  $c = \frac{7}{8}$  and  $s = \frac{53}{64}$ . This result became the new record for the upper bound of the approximation ratio of Min-2Lin(2)-deletion, as seen in Figure 3. Three years later, Wiman [16] further improved on this result by using Max-Had<sub>4</sub> instead of Max-Had<sub>3</sub>. Wiman’s Max-Had<sub>4</sub> gadget has  $c = \frac{15}{16}$  and  $s = \frac{3308625759}{3640066048} \approx 0.9089$ . This further improved the upper bound on the approximation ratio of Min-2Lin(2)-deletion.

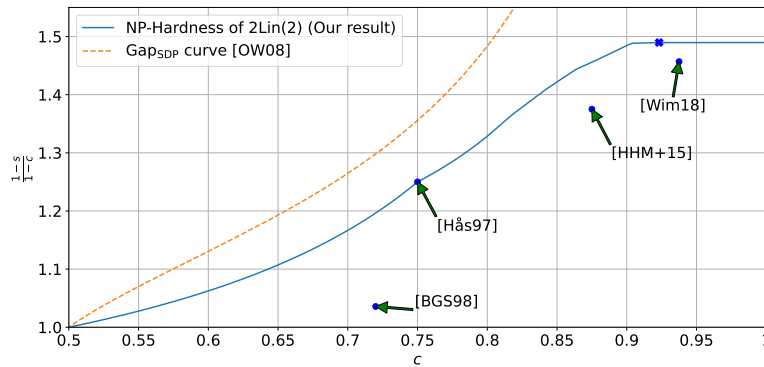
Similar to earlier results, the technique we use to construct our curve is also a gadget reduction from Max-Had <sub>$k$</sub>  to Max-2Lin(2). But instead of using a gadget reduction from Max-Had <sub>$k$</sub>  for a fixed  $k$ , we instead let  $k$  tend to infinity. This improves the quality of our gadget. One example of such an improvement is our upper bound on the approximation ratio of Min-2Lin(2)-deletion, which can be seen in Figure 3. The ratio  $\frac{1-s(c)}{1-c}$  is maximised on our curve at  $c = \frac{590174949}{639271832} \approx 0.9232$  and  $s = \frac{141533171}{159817958} \approx 0.8856$ , which is marked by a blue cross in Figure 3.



■ **Figure 1** The  $y$ -axis shows the soundness  $s$  and the  $x$ -axis the completeness  $c$ . The blue filled curve is our NP-hardness curve  $s(c)$ . The red dashed curve is the  $\text{Gap}_{\text{SDP}}(c)$  by O’Donnell and Wu’s [14]. The points marked with arrows are prior inapproximability results of Max-2Lin(2). The blue cross on the curve marks our best inapproximability result for Min-2Lin(2)-deletion, see Figure 3. Note that both of the curves in this figure are convex functions.



■ **Figure 2** The  $y$ -axis shows  $s/c$ , which corresponds to the approximation ratio of Max-2Lin(2). The point on the curve  $c(s)$  that minimises this ratio is  $c = 3/4$  and  $s(c) = 11/16$ , which exactly matches Håstad’s result from 1997 [9].



■ **Figure 3** The  $y$ -axis shows  $(1 - s)/(1 - c)$ , which corresponds to the approximation ratio of Min-2Lin(2)-deletion. This ratio reaches its maximum  $\frac{1-s(c)}{1-c} = \frac{73139148}{49096883} \approx 1.4896$  at  $c = \frac{590174949}{639271832}$ , which is marked by a blue cross. The curve stays constant after this point.

## 1.6 The limitations of $\text{Had}_k\text{-to-2Lin}(2)$ gadget reductions

In Figure 1, it is possible to see a clear gap between our  $s(c)$  curve and O’Donnell and Wu’s  $\text{Gap}_{\text{SDP}}(c)$  curve [14]. The gap is especially noticeable in Figure 3, since the behaviour of the two curves are completely different when  $c$  is close to 1. One natural question is, how close can a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget reduction get to the  $\text{Gap}_{\text{SDP}}(c)$  curve?

Håstad et al [10] showed that any gadget reduction from a Hadamard Max-CSP to Max-2Lin(2) can never achieve an approximation ratio for Min-2Lin(2)-deletion better than  $\frac{1}{1-e^{-0.5}} \approx 2.54$ . In Appendix B we show that any gadget reduction from a Hadamard CSP to Max-2Lin(2) that uses Wiman’s soundness relaxation can never achieve an approximation ratio of Min-2Lin(2)-deletion better than 2. Both 2.54 and 2 are fairly large in comparison to the current best value of 1.48969 shown in Figure 3. So it is potentially possible to still improve our results in the future using a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget reduction for some  $k \geq 4$ . However, these limitations means that it is impossible to make  $s(c)$  match the behaviour of  $\text{Gap}_{\text{SDP}}(c)$  when  $c$  is close to 1.

## 1.7 Outline of proof

Our result is based on  $\text{Had}_k\text{-to-2Lin}(2)$  gadget reduction for arbitrary large values of  $k$ . We start from the “automated gadget” framework by Trevisan et al [15]. In this framework, computing the soundness of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget involves solving a Max-2Lin(2) problem. Following the work of Wiman [16], we relax the soundness computation to a Max-Flow problem on the  $2^k$ -dimensional hypercube. Using symmetries, it is computationally feasible to construct  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets that are optimal with respect to the relaxed soundness for  $k \leq 4$ .

In order to be able harness the power of arbitrarily large  $k$ , we define a procedure of embedding a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $G$  inside a  $\text{Had}_{k'}\text{-to-2Lin}(2)$  gadget where  $k' > k$ . By overlapping multiple different embeddings of  $G$ , we construct a gadget  $G'$  for an arbitrarily large  $k'$ .

Recall that the relaxed soundness computation is a Max-Flow problem, which can be expressed as an LP. By carefully relaxing this LP, we are able to create an infeasible flow solution to  $\text{rs}(G)$ , such that if we lift it, it becomes an almost feasible flow of  $\text{rs}(G')$ . The underlying idea for this relaxation is based on leaky flows (flows where the flow entering a node can be different than the flow exiting the node). The “leaks” of a leaky flow are signed, so random overlap of leaky flows can result in a feasible flow. We show that this is actually the case for the solution to our relaxed LP using a second order moment analysis.

The final step is to construct the  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $G$  and its corresponding leaky flow for  $k = 4$  used in the embedding. This construction is naturally done using a rational LP solver to solve the relaxed LP.

## 1.8 Organisation of paper

Section 2 contain the preliminaries. It introduces Max-CSPs and the automated gadget framework. Section 3 introduces Wiman’s relaxed soundness and the infinity relaxed soundness in terms of an LP. This section also states our main Lemma, Lemma 3.11, relating the infinity relaxed soundness to the relaxed soundness. Appendix A is about Max-Flow, and it proves some general theorems about how symmetries can be used to simplify Max-Flow problems. Appendix B contain an analysis of relaxed soundness, and how it relates to the (true) soundness. Appendix C studies affine maps. These affine maps are used both to analyse the infinity relaxed soundness, and to describe the symmetries of the LPs. Appendix D



contains the proof of Lemma 3.11 using the affine maps. Appendix E describes the procedure we use for constructing and verifying the gadgets. Section 4 contains our numerical results. This includes both plots and tables of various  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets. Finally, Appendix F contains a compact description of all gadgets that we construct.

## 2 Preliminaries

This section is split into three parts. In Subsection 2.1 we introduce some basic concepts and notations for Boolean functions  $\mathbb{F}_2^k \rightarrow \{1, -1\}$ . After that, in Subsection 2.2 we formally define the  $(2^k - 1)$ -ary Hadamard predicate. The last subsection, Subsection 2.3, introduces the “automated gadget” framework by Trevisan et al [15], and explains the classical result of how to construct reductions from the  $(2^k - 1)$ -ary Hadamard predicate to  $\text{Max-2Lin}(2)$ .

### 2.1 Boolean functions

A  $k$ -bit Boolean function is a function that takes in  $k$  bits and outputs one bit. The  $k$  input bits should be thought of as a vector in a  $k$ -dimensional vector field over  $\mathbb{F}_2$ . On the other hand, the output bit is a scalar. For convenience, we denote the vectors as being elements in  $\mathbb{F}_2^k$  and the scalars as elements in  $\mathbb{R}$ , where a scalar bit is represented as 1 (False) or  $-1$  (True).

► **Definition 2.1.** *The set of  $k$ -bit Boolean functions is denoted by  $\mathcal{F}_k = \{f : \mathbb{F}_2^k \rightarrow \{1, -1\}\}$ .*

One special type of Boolean functions that is of great importance is the set of linear Boolean functions. Each linear Boolean function in  $\mathbb{F}_2^k$  corresponds to an element  $\alpha \in \mathbb{F}_2^k$ , and is denoted by  $\chi_\alpha$ .

► **Definition 2.2.** *For  $\alpha \in \mathbb{F}_2^k$  let  $\chi_\alpha \in \mathcal{F}_k$  be denote the function*

$$\chi_\alpha(x) = (-1)^{(\alpha, x)}$$

where  $(\alpha, x) = \sum_{i=1}^k \alpha_i x_i \pmod{2}$ .

Any Boolean function can be represented as a sum of linear Boolean functions using the Fourier transform.

► **Proposition 2.3.** *Given  $f \in \mathcal{F}_k$ , then*

$$f(x) = \sum_{\alpha \in \mathbb{F}_2^k} \chi_\alpha(x) \hat{f}_\alpha,$$

where  $\hat{f}_\alpha$  denotes the Fourier transform of  $f$  at  $\alpha$ , defined as

$$\hat{f}_\alpha = \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} \chi_\alpha(x) f(x), \quad \alpha \in \mathbb{F}_2^k.$$

The Fourier transform is used to define the supporting affine subspace of a Boolean function. This also gives a natural definition for the dimension of a Boolean function.

► **Definition 2.4.** *Given  $f \in \mathcal{F}_k$ , its supporting affine sub-space  $\text{affine}(f)$  is the affine span of  $\{\alpha \in \mathbb{F}_2^k : \hat{f}_\alpha \neq 0\}$ .*

► **Definition 2.5.** *Let  $\dim(f)$ ,  $f \in \mathcal{F}_k$ , denote the dimension of  $\text{affine}(f)$ .*

## 11:10 On the NP-Hardness Approximation Curve for Max-2Lin(2)

► Remark 2.6. Affine functions have dimension 0.

The distance between two Boolean function is given by the normalised Hamming distance.

► **Definition 2.7.** Let  $\text{dist} : \mathcal{F}_k \times \mathcal{F}_k \rightarrow \mathbb{R}$  be the normalised Hamming distance between two Boolean functions, i.e.

$$\text{dist}(f_1, f_2) = \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} \frac{1 - f_1(x)f_2(x)}{2}.$$

## 2.2 Max-CSP

This section introduces Constraint Satisfaction Problems (CSP) and Max-CSP. The framework we use is that CSPs are defined by predicates, which describe which kind of constraints that can appear in the CSP.

► **Definition 2.8.** An  $m$ -ary predicate is a function  $\phi \in \mathcal{F}_m$ . The predicate is said to be satisfied by  $x \in \mathcal{F}_2^m$  if  $\phi(x) = -1$ . Otherwise  $x$  is said to violate  $\phi$ . The set of  $x \in \mathcal{F}_2^m$  that satisfies  $\phi$  is denoted by  $\text{Sat}(\phi)$ .

Given a set of Boolean variables  $V$  and a  $m$ -ary predicate  $\phi$ , a  $\phi$ -constraint  $\mathcal{C}$  is a tuple  $((x_1, b_1), \dots, (x_m, b_m))$  where  $x_i \in V, i = [m]$ , and  $b_i \in \mathbb{F}_2, i \in [m]$ , where all of the  $x_i$ 's are distinct. The constraint  $\mathcal{C}$  is said to be satisfied if

$$\phi(b_1 + x_1, \dots, b_m + x_m) = -1,$$

where  $+$  denotes the xor-operation. In other words, if  $b_i = 1$  then  $x_i$  is negated.

► **Definition 2.9.** Given a  $m$ -ary predicate  $\phi$ , an instance  $\mathcal{I}$  of the Max- $\phi$ -CSP is a variable set  $V$  and a distribution of  $\phi$ -constraints over  $V$ . The Max- $\phi$ -CSP optimisation problem is; given an instance  $\mathcal{I}$ , find the assignment  $A : V \rightarrow \mathbb{F}_2$  that maximises the fraction of satisfied constraints in  $\mathcal{I}$ . The optimum is called the value of  $\mathcal{I}$ .

The main Max-CSPs of interest in this paper are the Hadamard  $\text{Had}_k$  Max-CSP, and Max-2Lin(2) and Max-3Lin(2). These have the following predicates.

► **Definition 2.10.** The 2Lin(2) predicate is the function  $f(x, y) = (-1)^{x+y+1}$ . Similarly, the 3Lin(2) predicate is the function  $f(x, y, z) = (-1)^{x+y+z+1}$ .

► **Definition 2.11.** The Hadamard  $\text{Had}_k$  predicate for  $k \geq 2$  is a  $(2^k - 1)$ -ary predicate. There is one input variable per non-empty subset  $S \subseteq [k]$ . The  $\text{Had}_k$  predicate is satisfied by a binary input string  $\{x_S\}_{\emptyset \neq S \subseteq [k]}$  if and only if there exists some  $\beta \subseteq [k]$  such that

$$\chi_\beta(S) = (-1)^{x_S}$$

for all non-empty subset  $S \subseteq [k]$ . I.e. the  $\text{Had}_k$  predicate is satisfied if and only if the input string forms the truth table of a linear function.

► Remark 2.12. The 3Lin(2) predicate and the  $\text{Had}_2$  predicate are in fact identical. Thus the family of Hadamard Max-CSPs can be seen as a natural generalisation of Max-3Lin(2).

► Remark 2.13. The set  $\text{Sat}(\text{Had}_k \text{ predicate})$  can be expressed using a  $2^k$  dimensional Hadamard matrix. Let  $M_k$  be a  $2^k \times (2^k - 1)$  matrix, where the rows are index by subsets  $\beta \subseteq [k]$  and the columns are indexed by non-empty subsets  $S \subseteq [k]$ . Let

$$(M_k)_{\beta, S} = \begin{cases} 0 & \text{if } \chi_\beta(S) = 1, \\ 1 & \text{if } \chi_\beta(S) = -1. \end{cases}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

■ **Figure 4** The matrix  $M_k$  for  $k = 3$ . It is an  $8 \times 7$  matrix. Note that prepending a zero column to  $M_k$  and switching 0/1 to  $1/-1$  would make it into a Hadamard matrix, which is symmetric.

The matrix  $M_k$  is the set  $\text{Sat}(\text{Had}_k \text{ predicate})$  expressed on the form of a matrix, with one row per element. Note that  $M_k$  is almost the  $2^k$ -dimensional Hadamard matrix.  $M_k$  can be made into the Hadamard matrix by prepending an all 0 column to it, and then switching out 0/1 to  $1/-1$ . This connection between  $\text{Sat}(\text{Had}_k \text{ predicate})$  and Hadamard matrices is one of the reasons as to why this Max-CSP is called the Hadamard Max-CSP. An example of the matrix  $M_k$  can be found in Figure 4.

The Hadamard predicate has been shown to be a *useless predicate*. The concept of useless predicates was first introduced in [3]. This property of  $\text{Had}_k$  was originally proven by Austrin and Mossel using UGC [4], which relies on the fact that  $\text{Sat}(\text{Had}_k)$  admits a balanced pairwise independent set. Later on Chan was able to show that  $\text{Had}_k$  is a useless predicate without requiring UGC [6]. To state this result we first need two definitions.

► **Definition 2.14.** Given an instance  $\mathcal{I}$  of an  $m$ -ary Max-CSP and an assignment  $A$ , let  $\mathcal{D}(A, \mathcal{I})$  denote the distribution of binary strings  $\mathbb{F}_2^m$  generated by sampling  $((x_1, b_1), \dots, (x_m, b_m)) \sim \mathcal{I}$  and outputting the binary string  $((A(x_1) + b_1), \dots, (A(x_m) + b_m))$ .

► **Definition 2.15.** The total variation distance  $d_{\text{TV}}$  between two probability measures  $\mu_1$  and  $\mu_2$  over a finite set  $\Omega$  is defined as

$$d_{\text{TV}}(\mu_1, \mu_2) = \frac{1}{2} \sum_{\omega \in \Omega} |\mu_1(\omega) - \mu_2(\omega)|.$$

► **Theorem 2.16** ([6]). For every  $\varepsilon > 0$ , it is NP-hard to distinguish between instances  $\mathcal{I}$  of the  $\text{Had}_k$  Max-CSP such that

**Completeness** There exists an assignment  $A$  such that

$$d_{\text{TV}}(\mathcal{D}(A, \mathcal{I}), \text{uniform}(\{\text{Sat}\}(\text{Had}_k \text{ predicate}))) \leq \varepsilon.$$

**Soundness** For every assignment  $A$ ,

$$d_{\text{TV}}(\mathcal{D}(A, \mathcal{I}), \text{uniform}(\mathbb{F}_2^{2^k - 1})) \leq \varepsilon.$$

Here  $\text{uniform}(\text{Sat}(\text{Had}_k \text{ predicate}))$  denotes the uniform distribution over binary strings that satisfy the  $\text{Had}_k$  predicate. Similarly,  $\text{uniform}(\mathbb{F}_2^{2^k - 1})$  denotes the uniform distribution over all binary strings of length  $2^k - 1$ .

► **Remark 2.17.** The uniform distribution on satisfiable instances in the completeness case is a subtle detail. The result by [6] is not formulated like this. However, it is trivial to take the instances constructed by Chan and modify them to make the completeness case be uniformly

distributed over satisfied instances. The first time this was used was by [16]. However, this detail turns out to not actually matter in the end since all of the gadgets that we construct and all of the gadgets that Wiman construct are symmetric. So this uniform randomness assumption is only there because of convenience, and is not actually used in the end.

### 2.3 The automated gadget framework

The “automated gadget” framework by Trevisan et al [15] describes how to construct optimal gadgets when reducing from one predicate to another. Let us denote the starting predicate as  $\phi$  and the target predicate as  $\psi$ . A  $\phi$ -to- $\psi$ -gadget is a description for how to reduce a  $\phi$ -constraint to one or more  $\psi$ -constraints. As an example, let us take a gadget from 3SAT to 2Lin(2). In this case the gadget describes a system of linear equations that both involve the three original variables from the 3SAT constraint (called *primary variables*, denoted by  $\mathbb{X}$ ) as well as new extra variables (called *auxiliary variables*, denoted by  $\mathbb{Y}$ ).

Gadgets have two important properties, called *completeness* and *soundness*. These properties describe how closely the  $\psi$ -constraints are able to mimic the satisfiability of the original  $\phi$ -constraint. The completeness of a gadget is a value between 0 and 1 that describe how many of the  $\psi$ -constraints that can be satisfied under the restriction that  $\mathbb{X}$  satisfies the original  $\phi$ -constraint. In a similar fashion, the soundness of a gadget is a value between 0 and 1 that describes the case when  $\mathbb{X}$  does not satisfy the original  $\phi$ -constraint. When we construct our gadgets, we fix the completeness of the gadget, and then we find the gadget that minimises the soundness for this fixed completeness. A gadget that minimises the soundness for a given completeness is referred to as an *optimal gadget*.

There is no a priori upper bound on how many auxiliary variables that a  $\phi$ -to- $\psi$ -gadget can have. However, the “automated gadget” framework by Trevisan et al [15] proves that, under some reasonable assumptions, the number of variables  $|\mathbb{X} \cup \mathbb{Y}|$  in an optimal gadget can be assumed to be at most  $2^{|\text{Sat}(\phi)|}$ . Furthermore, if  $\psi$  allows the negations of variables, then this number drops to  $2^{|\text{Sat}(\phi)|-1}$ .

In the case of a Had<sub>k</sub>-to-2Lin(2) gadget, the number of satisfying assignments of Had<sub>k</sub> is  $2^k$ , and 2Lin(2) allow the negation of variables. This means that the total number of variables in the gadget is  $2^{2^k-1}$ . Out of these,  $2^k - 1$  variables are in  $\mathbb{X}$ , and  $2^{2^k-1} - (2^k - 1)$  variables are in  $\mathbb{Y}$ . Furthermore, the “automated gadget” framework gives a natural way to index these variables in terms of  $|\text{Sat}(\text{Had}_k)|$ -long bitstrings. According to the framework, each primary variable should be indexed by a bitstring describing that variable’s assignment to all of the satisfying assignments to Had<sub>k</sub>, meaning a column in the matrix shown in Figure 4. The auxiliary variables are indexed by the bitstrings that do not appear in the matrix.

Instead of using  $2^k$ -long bitstrings to index the variables, it is arguably more natural to index the variables using functions in  $\mathbb{F}_2^k$ . These representations are equivalent since every  $2^k$  long bitstring can be interpreted as a truth table of a function in  $\mathcal{F}_k$ , and vice versa. By indexing the set of variables using functions in  $\mathcal{F}_k$ , the set of primary variables are indexed by linear functions  $\{\chi_\alpha\}_{\emptyset \subset \alpha \subseteq [k]}$ , and the negations of linear functions  $\{-\chi_\alpha\}_{\emptyset \subset \alpha \subseteq [k]}$ . This gives us the following description of a Had<sub>k</sub>-to-2Lin(2) gadget.

► **Definition 2.18.** A Had<sub>k</sub>-to-2Lin(2) gadget is given by a tuple  $(G, \mathbb{X}_k, \mathbb{Y}_k)$ , where  $G$  is a probability distribution over  $\binom{\mathcal{F}_k}{2}$  where  $G(f_1, f_2) = 0$  if  $f_1 = -f_2$ .  $\mathbb{X}_k$  is the set of primary variables and  $\mathbb{Y}_k$  is the set of auxiliary variables. The set of variables  $\mathbb{X}_k \cup \mathbb{Y}_k$  are indexed by functions in  $\mathcal{F}_k$ , meaning  $\mathbb{X}_k \cup \mathbb{Y}_k = \{x_f : f \in \mathcal{F}_k\}$ . A variable  $x_f$  is a primary variable if and only if  $f$  is a linear function or  $-f$  is a linear function.

The reduction from a  $\text{Had}_k$  constraint  $\text{Had}_k(b_{\{1\}} + y_{\{1\}}, \dots, b_{\{k\}} + y_{\{k\}})$  to  $2\text{Lin}(2)$  is given by the distribution formed by

1. Sampling  $(f_1, f_2) \sim G$ ,
2. Outputting the constraint  $T(f_1) = T(f_2)$  where

$$T(f) = \begin{cases} x_f & \text{if } x_f \in \mathbb{Y}_k, \\ b_\alpha + y_\alpha & \text{if } f = \chi_\alpha \text{ for some } \alpha \in \mathbb{F}_2^k, \\ b_\alpha + y_\alpha + 1 & \text{if } f = -\chi_\alpha \text{ for some } \alpha \in \mathbb{F}_2^k. \end{cases}$$

Let us now precisely define the soundness and completeness of a  $\text{Had}_k$ -to- $2\text{Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$ . From Theorem 2.16 it follows that the natural definition of soundness is to uniformly at random assign the primary variables  $\mathbb{X}_k$  to  $\mathbb{F}_2$ , and then assign the rest of the variables in order to satisfy as many of the equations as possible.

► **Definition 2.19.** Given a set of Boolean variables  $\mathbb{X}$ . Let  $\mathcal{F}(\mathbb{X})$  denote the set of assignments  $\mathbb{X} \rightarrow \mathbb{F}_2$ . Let  $\mathcal{F}_{\text{fold}}(\mathbb{X})$  the set of all folded assignments, i.e. functions  $P : \mathbb{X} \rightarrow \mathbb{F}_2$  such that  $P(1+x) = 1 + P(x) \forall x \in \mathbb{X}$ . Here  $1+x$  denotes the negation of the variable  $x$ .

► **Definition 2.20.** The soundness of  $G$  is defined as

$$s(G) = \mathbb{E}_{P \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k)} \max_{\substack{A \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \mathbb{Y}_k), \\ A(x) = P(x), x \in \mathbb{X}_k}} \text{val}(A, G),$$

where

$$\text{val}(A, G) = \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2)[A(x_{f_1}) = A(x_{f_2})].$$

The completeness of  $G$  is defined using dictator cuts. A *dictator cut*  $\delta_y$  of  $y \in \mathbb{F}_2^k$  is an assignment where  $(-1)^{\delta_y(x_f)} = f(y)$ . From Theorem 2.16 we see that the natural definition for completeness is the expectation over  $\text{val}(\delta_y, G)$ , where  $\delta_y$  is a random dictator cut.

► **Definition 2.21.** The completeness of  $G$  is defined as

$$c(G) = \mathbb{E}_{y \in \mathbb{F}_2^k} \text{val}(\delta_y, G) = 1 - \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2) \text{dist}(f_1, f_2).$$

There is a result based on Theorem 2.16 that relates the soundness and completeness of  $\text{Had}_k$ -to- $2\text{Lin}(2)$  gadgets to NP-hardness results for  $\text{Max-2Lin}(2)$ .

► **Proposition 2.22** ([10, Proposition 2.17]). Given a  $\text{Had}_k$ -to- $2\text{Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  with  $s = s(G)$  and  $c = c(G)$ , where  $c > s$ . Then for every sufficiently small  $\varepsilon > 0$ , it is NP-hard to distinguish between instances  $\mathcal{I}$  of  $\text{Max-2Lin}(2)$  such that

(Completeness) There exists an assignment that satisfies a fraction at least  $c - \varepsilon$  of the constraints.

(Soundness) All assignments satisfy at most a fraction  $s + \varepsilon$  of the constraints.

One particularly interesting case is the inapproximability of  $\text{Min-2Lin}(2)$ -deletion. From UGC it follows that it is NP-hard to approximate  $\text{Min-2Lin}(2)$ -deletion within any constant [14]. The following proposition from Håstad et al. [10] tells us that a  $\text{Had}_k$ -to- $2\text{Lin}(2)$  gadget reduction can never be used to show an inapproximability factor of  $\text{Min-2Lin}(2)$ -deletion better than 2.54. This means that any NP-hardness result for  $\text{Min-2Lin}(2)$ -deletion shown using a gadget reduction from  $\text{Had}_k$ -to- $2\text{Lin}(2)$  cannot match results obtained by UGC.

## 11:14 On the NP-Hardness Approximation Curve for Max-2Lin(2)

► **Proposition 2.23** ([10, Proposition 2.29 and Theorem 6.1]). *For any given  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$ . There exists a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$  with completeness  $1 - 2^{-k}$  such that*

$$\frac{1 - s(G)}{1 - c(G)} \leq \frac{1 - s(\tilde{G})}{1 - c(\tilde{G})},$$

and

$$\frac{1 - s(\tilde{G})}{1 - c(\tilde{G})} \leq \frac{1}{1 - e^{-0.5}} \approx 2.54.$$

► **Remark 2.24.** A  $\text{Had}_k\text{-to-2Lin}(2)$  gadget having completeness  $1 - 2^{-k}$  implies that the gadget only have positive weight edges of length  $2^{-k}$ . So far fewer edges are used compared to the total number of possible edges.

► **Remark 2.25.** The upper limit of 2.54 shown by [10] is much more general than what is stated here. In fact, they show that the bound of 2.54 holds for any gadget reduction from a useless predicate  $\phi$  such that  $\text{Sat}(\phi)$  has a balanced pairwise independent subset.

### 3 Relaxed soundness and infinity relaxed soundness

The main difficulty when designing and analysing gadgets is that the soundness is difficult to compute. In the case of a gadget reduction from Max-Had $_k$  to Max-2Lin(2), computing the soundness of the gadget involves solving an instance of Max-2Lin(2). For  $k \leq 3$  this is computationally feasible, since the Max-2Lin(2) instance is rather small, but for  $k \geq 4$  the instances can become so large that, even using a computer, it is practically impossible to solve them.

To get around this issue, Wiman [16] proposed to relax the definition of the soundness by not requiring that the assignment  $A$  of the auxiliary variables  $\mathbb{Y}_k$  is folded. Note that the assignment  $A$  is still required to be folded on the primary variables  $\mathbb{X}_k$ , meaning  $A(x_f) = 1 + A(x_{-f}) \forall x_f \in \mathbb{X}_k$ . Removing the requirement that  $A$  is folded over  $\mathbb{Y}_k$  makes it significantly easier to compute the soundness.

► **Definition 3.1** ([16, Definition 3.3]). *Wiman's relaxed soundness*

$$\text{rs}(G) = \max_{\substack{P \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \{x_1, x_{-1}\}) \\ A \in \mathcal{F}(\mathbb{X}_k \cup \mathbb{Y}_k), \\ A(x) = P(x), x \in \mathbb{X}_k \cup \{x_1, x_{-1}\}}} \mathbb{E} \text{val}(A, G),$$

where

$$\text{val}(A, G) = \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2)[A(x_{f_1}) = A(x_{f_2})].$$

This relaxation fundamentally changes the soundness computation from being a Max-2Lin(2) problem to being an  $s$ - $t$  Min-Cut problem. This is because the computation of  $1 - \text{rs}(G)$  for a fixed  $P$  is a minimisation problem where the goal is to minimise the number of times that  $A(x_{f_1}) \neq A(x_{f_2})$ , which makes it a  $s$ - $t$  Min-Cut problem. According to the Max-Flow Min-Cut Theorem, this also means that  $\text{rs}(G)$  can be computed by solving a Max-Flow problem.

The conclusion from this is that  $1 - \text{rs}(G)$  can be interpreted as the average max flow on the fully connected  $2^k$ -dimensional hypercube, where the placement of sources and sinks is randomly distributed over nodes labeled by affine functions. The sources correspond to

primary variables  $x_f$  where  $P(x_f) = 1$  and the sink nodes correspond to primary variables  $x_f$  where  $P(x_f) = 0$ . The capacity of an edge  $\{x_{f_1}, x_{f_2}\}$  in the fully connected hypercube is given by  $G(f_1, f_2)$ . Note that the sum over capacities in the graph is equals to 1.

There are some significant benefits to using the relaxed soundness. Firstly, it is significantly simpler to solve a Max-Flow problem compared to a Max-2Lin(2) problem. The implication from this is that it is computationally simple to compute the relaxed soundness of Had<sub>4</sub>-to-2Lin(2) gadgets and even possible to compute relaxed soundness of Had<sub>5</sub>-to-2Lin(2) gadgets if given enough computational resources. Furthermore, the relaxed soundness allows us to analyse Had<sub>k</sub>-to-2Lin(2) gadgets even in the case where  $k$  is very large. The disadvantage to using relaxed soundness is that it is not guaranteed to be close to the true soundness.

### 3.1 Relaxed soundness described as an LP

Recall that  $1 - \text{rs}(G)$  can be expressed as the average max flow on a fully connected  $2^k$ -dimensional hypercube with randomised source/sink placements. This means that  $\text{rs}(G)$  can be stated as an LP. One reason for why it is preferable to express this Max-Flow problem as an LP is because it is possible to move the capacities (i.e. the ‘‘gadget variables’’) of the Max-Flow problem to the variable side of the LP. So the same LP can be used both to calculate the the relaxed soundness of a specific gadget and to construct new gadgets.

One additional step we use in the formulation of this LP is to use a function  $g \in \mathcal{F}_k$  to describe the source/sink placement instead of using the assignment  $P$ . A node  $v_{\chi_\alpha}$ ,  $\alpha \in \mathbb{F}_2^k$  is a sink node if  $g(\alpha) = 1$ , and a source node if  $g(\alpha) = -1$ . These two representations of the source/sink placement are equivalent, but using a Boolean function  $g$  is more helpful for understanding the symmetries of the LP, as done in Appendix C. The following is the LP reformulation of the relaxed soundness  $\text{rs}(G)$ .

► **Definition 3.2.** A flow  $w$  of a Had<sub>k</sub>-to-2Lin(2) gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  is a function  $\mathcal{F}_k^3 \rightarrow \mathbb{R}_{\geq 0}$ . The flow  $w$  is said to be feasible if and only if

$$w(f_1, f_2, g) + w(f_2, f_1, g) \leq G(f_1, f_2) \quad \forall f_1, f_2, g \in \mathcal{F}_k, \quad (1)$$

$$\text{out}_w(f, g) = \text{in}_w(f, g) \quad \forall f, g \in \mathcal{F}_k, \dim(f) \geq 1. \quad (2)$$

where  $\text{out}_w(f, g) = \sum_{f_2 \in \mathcal{F}_k} w(f, f_2, g)$  and  $\text{in}_w(f, g) = \sum_{f_2 \in \mathcal{F}_k} w(f_2, f, g)$ . The value of  $w$  for at a source/sink placement  $g \in \mathcal{F}_k$  is defined as

$$\text{val}_g(w) = \sum_{\alpha \in \mathbb{F}_2^k} \text{out}_w(g(\alpha)\chi_\alpha, g) - \text{in}_w(g(\alpha)\chi_\alpha, g).$$

► **Definition 3.3.** The relaxed soundness LP for a Had<sub>k</sub>-to-2Lin(2) gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$ , denoted by  $\text{rsLP}(G)$ , is the following LP

$$\text{rs}(G) = 1 - \max_w \mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w),$$

where the maximum is taken over feasible flows  $w$  of  $G$ .

► **Remark 3.4.** Recall that  $1 - \text{rs}(G)$  is the average of  $2^{2^k}$  independent Max-Flow problems. The different Max-Flow problems are indexed by the function  $g \in \mathcal{F}_k$ , which describes the placements of sinks and sources. The nodes in each Max-Flow problem are indexed by functions in  $\mathcal{F}_k$ . The sink nodes in the  $g$ -th Max-Flow problem are the nodes  $v_{g(\alpha)\chi_\alpha}$ ,  $\alpha \in \mathbb{F}_2^k$ , and the source nodes are the nodes  $v_{-g(\alpha)\chi_\alpha}$ ,  $\alpha \in \mathbb{F}_2^k$ . The flow from  $v_{f_1} \rightarrow v_{f_2}$  is  $w(f_1, f_2, g)$ , and the capacity of the undirected edge  $\{v_{f_1}, v_{f_2}\}$  is  $G(f_1, f_2)$ .

► **Remark 3.5.** Note that it is possible to modify the  $\text{rsLP}(G)$  to include the capacities of the graph (i.e. gadget  $G$ ) as variables. The implications of this is that the optimisation problem of finding a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget with the maximum relaxed soundness for a fixed completeness can also be expressed as an LP.

► **Remark 3.6.** Note that the  $\text{rsLP}(G)$  has roughly  $|\mathcal{F}_k|^3 = 2^{3 \cdot 2^k}$  variables. This is a very large number, even for small values of  $k$ . So in order to be able to solve this LP, we have to make use of the symmetries of the LP in order to reduce the number of variables.

### 3.2 Introduction of infinity relaxed soundness

One natural question is, how small can one make the relaxed soundness if we fix the completeness of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget and let  $k \rightarrow \infty$ ? In practice, even just finding the gadget minimising the relaxed soundness when  $k = 5$  is a very daunting task, so cannot hope to calculate this limit directly from the  $\text{rsLP}(G)$ .

Our method to handle large values of  $k$  is to create a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget for some small value of  $k$ , for example  $k = 4$ , and then introduce the concept of embedding a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $G$  into a  $\text{Had}_{k'}\text{-to-2Lin}(2)$  gadget  $G'$ , where  $k' > k$ . It is also possible to embed the flow of the  $\text{rsLP}(G)$  onto the  $\text{rsLP}(G')$ . This embedding has the property that the completeness and the soundness of both gadgets are the same.

The key insight is that by using multiple overlapping embeddings of  $G$ , we can improve the soundness of  $G'$  without affecting its completeness. Our argument for why multiple overlapping embeddings improve the relaxed soundness is based on leaky flows. Note that the leaks of a leaky flow have signs. This means that it is possible that overlapping embeddings of leaky flows could become a feasible flow, since the overlap of the embeddings could cause the signed leaks to sum to 0. We use this type of argument to show an upper bound on  $\text{rs}(G')$  based on a leaky flow solution to the  $\text{rsLP}(G)$ .

The exact procedure for the embeddings is defined in Appendix C and analysed in detail in Appendix D using second moment analysis. The conclusion from that analysis is that the following relaxation of the  $\text{rsLP}(G)$ , which we call the *infinity relaxed soundness LP*, denoted by  $\text{rs}_\infty\text{LP}(G)$ , has the following two important properties. Firstly, the solution of  $\text{rs}_\infty\text{LP}(G)$  is a leaky flow of  $\text{rsLP}(G)$ , and secondly, overlapping embeddings of this leaky flow tends to a feasible flow of  $\text{rsLP}(G')$  as  $k' \rightarrow \infty$ .

► **Definition 3.7.** A flow  $\tilde{w}$  of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  is said to be a *infinity relaxed flow* if constraint (1) is satisfied and

$$\sum_{g'} \text{out}_w(f, g') = \sum_{g'} \text{in}_w(f, g') \quad \forall g, f \in \mathcal{F}_k : \dim(f) \geq 1, \quad (3)$$

where the sums are over functions  $g' \in \mathcal{F}'_k$  such that  $g'|_{\text{affine}(f)} = g|_{\text{affine}(f)}$ . The (signed) leak at  $(f, g)$ , where  $f, g \in \mathcal{F}_k, \dim(f) \geq 1$ , is defined as  $\text{leak}_{\tilde{w}}(f, g) = \text{in}_w(f, g) - \text{out}_w(f, g)$ .

► **Definition 3.8.** The *infinity relaxed soundness* of  $G$ , denoted by  $\text{rs}_\infty(G)$ , is the solution to the  $\text{rs}_\infty\text{LP}(G)$

$$\text{rs}_\infty(G) = 1 - \max_{\tilde{w}} \mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(\tilde{w}),$$

where the maximum is taken over all infinity relaxed flows  $\tilde{w}$  of  $G$ .

► **Remark 3.9.** The  $\text{rs}_\infty\text{LP}(G)$  is a constraint relaxation of the  $\text{rsLP}(G)$  where constraint (2) has been relaxed to constraint (3). So a solution of the  $\text{rs}_\infty\text{LP}(G)$  is a leaky flow in the  $\text{rsLP}(G)$ .



► **Remark 3.10.** Constraint (3) is used for a proof in Appendix D of Lemma D.5, which is a 2nd order moment analysis of the overlap of leaks from random embeddings. In the proof, constraint (3) is used to show that if  $w$  is an infinity relaxed flow then  $\forall g, f \in \mathcal{F}_k, \dim(f) \geq 1 : \sum_{g'} \text{leak}_w(f, g') = 0$ , where the sum is over  $g' \in \mathcal{F}_k$  such that  $g'|_{\text{affine}(f)} = g|_{\text{affine}(f)}$ .

The following lemma describes a relationship between the  $\text{rsLP}(G)$  and the  $\text{rs}_\infty\text{LP}(G)$ . This is the key Lemma, which is proven in Appendix D.

► **Lemma 3.11.** *Let  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  be a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget. For any  $\varepsilon > 0$  there exists a  $\text{Had}_{k'}\text{-to-2Lin}(2)$  gadget  $(G', \mathbb{X}_{k'}, \mathbb{Y}_{k'})$  for some  $k' > k$  such that  $c(G) = c(G')$  and  $\text{rs}(G') \leq \text{rs}_\infty(G) + \varepsilon$ .*

From this Lemma, it follows that  $\text{rs}_\infty(G) + \varepsilon$  is the upper bound of  $\text{rs}(G')$  for some gadget  $G'$ , which in turn is an upper bound of  $s(G')$ . This means that the NP-hardness result of  $\text{Max-2Lin}(2)$  stated in Proposition 2.22 for  $\text{rs}(G)$  also holds for  $\text{rs}_\infty(G)$ . This gives us our main result.

► **Theorem 3.12.** *Let  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  be a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget with  $s = \text{rs}_\infty(G)$  and  $c = c(G)$ , where  $c > s$ . Then for every sufficiently small  $\varepsilon > 0$ , it is NP-hard to distinguish between instances of  $\text{Max-2Lin}(2)$  such that*

**Completeness** *There exists an assignment that satisfies a fraction at least  $c - \varepsilon$  of the constraints.*

**Soundness** *All assignments satisfy at most a fraction  $s + \varepsilon$  of the constraints.*

## 4 Numerical results

This section contains a presentation of constructed  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets. Recall that there are three different ways to measure the soundness of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget. There is the true soundness of a gadget, which can be used to show NP-hardness results for  $\text{Max-2Lin}(2)$ , see Proposition 2.22. Then there is the relaxed soundness, denoted by  $\text{rs}$ . This is an upper bound of the true soundness, see Proposition B.1. Finally there is the infinity relaxed soundness, denoted by  $\text{rs}_\infty$ , which according to our main result, Theorem 3.12, also imply NP-hardness results for  $\text{Max-2Lin}(2)$ .

We compute gadgets for  $k = 2, 3, 4$ , optimised either for  $\text{rs}$  or  $\text{rs}_\infty$ . The short rundown of the process of constructing a gadget is to first decide on the completeness of the gadget, and then call an LP-solver to find the gadget with that completeness that either minimises  $\text{rs}$  or  $\text{rs}_\infty$ , depending on which measure of soundness we want to optimise the gadget for.

### 4.1 Edges used/unused in constructed gadgets

The capacity  $G$  of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$  is a probability distribution over (undirected) edges. Every gadget that we construct is symmetrical under the mappings of  $\mathcal{M}_{k \rightarrow k}$ , so edges from the same edge orbit share the same capacity. Tables 7–9 in Appendix F list all edge orbits that have non-zero weight in at least one of our constructed gadgets for  $k = 2, 3, 4$ . Note that as discussed in Appendix E.2.1, in the case of  $k = 4$  it is possible that the gadgets we construct are sub-optimal if  $c < 1 - 2^{-k}$ . This means that it is possible that the Table for  $k = 4$ , Table 9, could look slightly different had we constructed optimal gadgets.

## 11:18 On the NP-Hardness Approximation Curve for Max-2Lin(2)

■ **Table 1** The curve  $s(c)$  as shown in Figure 1. The values of  $s(c)$  in this table are rounded up to 4 decimals. This table has the same format as the table describing the  $\text{Gap}_{\text{SDP}}(c)$  curve, found in Appendix E of [14].

| $c$   | $s(c)$ | $c$   | $s(c)$ | $c$   | $s(c)$ | $c$   | $s(c)$ | $c$   | $s(c)$ |
|-------|--------|-------|--------|-------|--------|-------|--------|-------|--------|
| 0.500 | 0.5000 | 0.600 | 0.5750 | 0.700 | 0.6500 | 0.800 | 0.7343 | 0.900 | 0.8516 |
| 0.505 | 0.5038 | 0.605 | 0.5788 | 0.705 | 0.6538 | 0.805 | 0.7390 | 0.905 | 0.8586 |
| 0.510 | 0.5075 | 0.610 | 0.5825 | 0.710 | 0.6575 | 0.810 | 0.7437 | 0.910 | 0.8661 |
| 0.515 | 0.5113 | 0.615 | 0.5863 | 0.715 | 0.6613 | 0.815 | 0.7485 | 0.915 | 0.8735 |
| 0.520 | 0.5150 | 0.620 | 0.5900 | 0.720 | 0.6650 | 0.820 | 0.7535 | 0.920 | 0.8809 |
| 0.525 | 0.5188 | 0.625 | 0.5938 | 0.725 | 0.6688 | 0.825 | 0.7588 | 0.925 | 0.8884 |
| 0.530 | 0.5225 | 0.630 | 0.5975 | 0.730 | 0.6725 | 0.830 | 0.7642 | 0.930 | 0.8958 |
| 0.535 | 0.5263 | 0.635 | 0.6013 | 0.735 | 0.6763 | 0.835 | 0.7696 | 0.935 | 0.9032 |
| 0.540 | 0.5300 | 0.640 | 0.6050 | 0.740 | 0.6800 | 0.840 | 0.7752 | 0.940 | 0.9107 |
| 0.545 | 0.5338 | 0.645 | 0.6088 | 0.745 | 0.6838 | 0.845 | 0.7809 | 0.945 | 0.9181 |
| 0.550 | 0.5375 | 0.650 | 0.6125 | 0.750 | 0.6875 | 0.850 | 0.7868 | 0.950 | 0.9256 |
| 0.555 | 0.5413 | 0.655 | 0.6163 | 0.755 | 0.6922 | 0.855 | 0.7927 | 0.955 | 0.9330 |
| 0.560 | 0.5450 | 0.660 | 0.6200 | 0.760 | 0.6969 | 0.860 | 0.7988 | 0.960 | 0.9405 |
| 0.565 | 0.5488 | 0.665 | 0.6238 | 0.765 | 0.7016 | 0.865 | 0.8050 | 0.965 | 0.9479 |
| 0.570 | 0.5525 | 0.670 | 0.6275 | 0.770 | 0.7063 | 0.870 | 0.8115 | 0.970 | 0.9554 |
| 0.575 | 0.5563 | 0.675 | 0.6313 | 0.775 | 0.7109 | 0.875 | 0.8181 | 0.975 | 0.9628 |
| 0.580 | 0.5600 | 0.680 | 0.6350 | 0.780 | 0.7156 | 0.880 | 0.8247 | 0.980 | 0.9703 |
| 0.585 | 0.5638 | 0.685 | 0.6388 | 0.785 | 0.7203 | 0.885 | 0.8313 | 0.985 | 0.9777 |
| 0.590 | 0.5675 | 0.690 | 0.6425 | 0.790 | 0.7250 | 0.890 | 0.8380 | 0.990 | 0.9852 |
| 0.595 | 0.5713 | 0.695 | 0.6463 | 0.795 | 0.7297 | 0.895 | 0.8448 | 0.995 | 0.9926 |

## 4.2 Lists and plots of gadgets

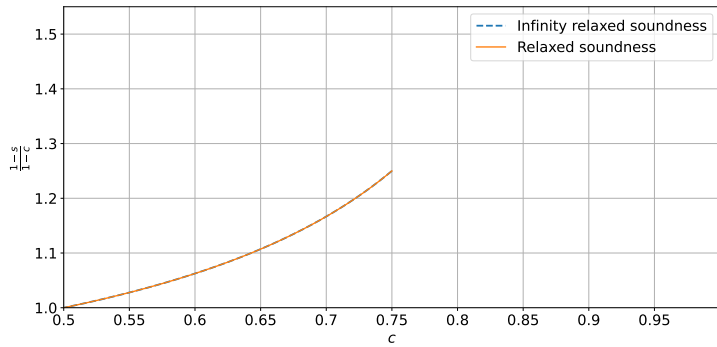
Figures 5, 6 and 7 show  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets with completeness on the  $x$ -axis, and either maximal  $\frac{1-\text{rs}(G)}{1-c(G)}$  or maximal  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$  on the  $y$ -axis. To create this plot, we construct one gadget for each completeness value from 0.5 to  $1 - 2^{-k}$  (inclusive), with a spacing of  $2^{-9}$ . The curve is constructed using interpolation by taking convex combinations of pairs of neighbouring gadgets.

### 4.2.1 The curve $s(c)$

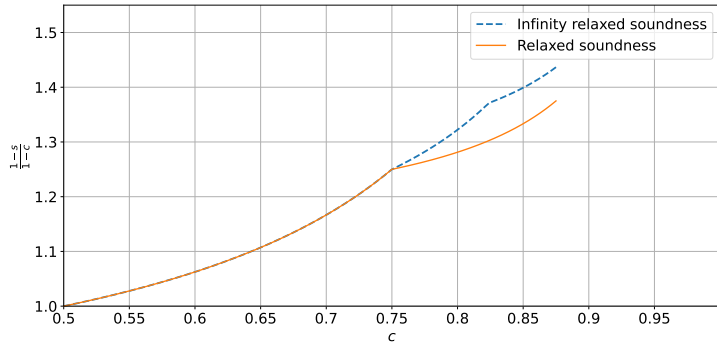
The curve  $s(c)$  describes the infinity relaxed soundness of  $\text{Had}_4\text{-to-2Lin}(2)$  gadgets as a function of completeness, shown as the upper curve in Figure 7, as well as in Figures 1, 2 and 3. The data for this curve can be found in Table 1. It has the following formal definition.

► **Definition 4.1.** *The curve  $s(c) : [0.5, 1] \rightarrow [0.5, 1]$ ,  $k = 4$ , is for  $c \in [0.5, 1 - 2^{-k}]$  defined as the solution to the restricted compressed  $\text{rs}_\infty\text{LP}$ . For  $c \geq 1 - 2^{-k}$  the curve is defined as  $s(c) = 1 + 2^k(s(1 - 2^{-k}) - 1)(1 - c)$ , meaning  $\frac{1-s(c)}{1-c}$  is constant for all  $c \geq 1 - 2^{-k}$ .*

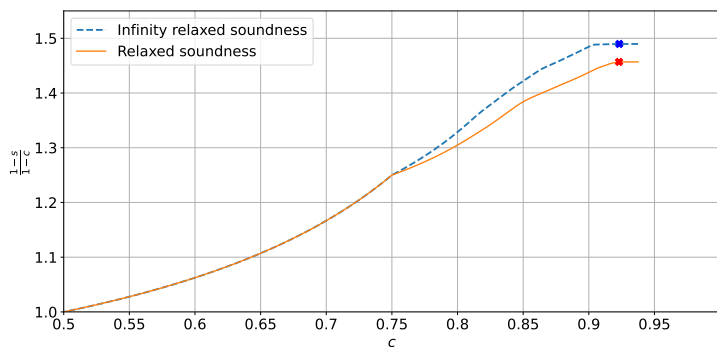
**Proof of Theorem 1.1.** For  $c \in [0.5, 1 - 2^{-k}]$ , the NP-hardness result follows directly from Theorem 3.12 since the solution of the restricted compressed  $\text{rs}_\infty\text{LP}(G)$  is an upper bound of the (non-restricted)  $\text{rs}_\infty\text{LP}(G)$ . For  $c \geq 1 - 2^{-k}$  the NP-hardness result follows from taking the convex combination of  $(c, s) = (1 - 2^{-k}, s(1 - 2^{-k}))$  and  $(c, s) = (1, 1)$ . Since it is possible to create a hard instance by taking the convex combination of two hard instances using separate variables. ◀



■ **Figure 5** This plot shows two types of Had<sub>2</sub>-to-2Lin(2) gadgets. The filled curve describes the minimisation of  $rs$  and the striped curve describes the minimisation of  $rs_\infty$ . The completeness value is on the  $x$ -axis, and either  $\frac{1-rs(G)}{1-c(G)}$  or  $\frac{1-rs_\infty(G)}{1-c(G)}$  on the  $y$ -axis. In this particular case, the case of  $k = 2$ , it turns out that these two curves are identical.



■ **Figure 6** This plot shows two types of Had<sub>3</sub>-to-2Lin(2) gadgets. The filled curve describes the minimisation of  $rs$  and the striped curve describes the minimisation of  $rs_\infty$ . The completeness value is on the  $x$ -axis, and either  $\frac{1-rs(G)}{1-c(G)}$  or  $\frac{1-rs_\infty(G)}{1-c(G)}$  on the  $y$ -axis.



■ **Figure 7** This plot shows two types of Had<sub>4</sub>-to-2Lin(2) gadgets. The filled curve describes the minimisation of  $rs$  and the striped curve describes the minimisation of  $rs_\infty$ . The completeness value is on the  $x$ -axis, and either  $\frac{1-rs(G)}{1-c(G)}$  or  $\frac{1-rs_\infty(G)}{1-c(G)}$  on the  $y$ -axis. The top part of both of these curves are perfectly flat, which is not the case in Figure 5 and Figure 6. The gadgets that mark the point where the curves become flat can be found in Tables 2 and 3, and are marked by crosses in the plot.

## 11:20 On the NP-Hardness Approximation Curve for Max-2Lin(2)

■ **Table 2** The Had<sub>4</sub>-to-2Lin(2) gadget  $G$  with minimal completeness among those that minimise  $\frac{1-\text{rs}(G)}{1-c(G)}$ . The completeness of  $G$  is  $c(G) = 9939/10768$  and relaxed soundness is  $\text{rs}(G) = 2623643487/2955083776$ . The right most column tells how much of the total capacity is contained in each edge orbit. This column sums up to 100%.

| $f_1$            | $f_2$            | length | $G(f_1, f_2)$    | % of total |
|------------------|------------------|--------|------------------|------------|
| 1100000000000000 | 1110000000000000 | 1      | 5461/969636864   | 30.3       |
| 1110000000000000 | 1111000000000000 | 1      | 17007/1616061440 | 18.9       |
| 1110000000000000 | 1110100000000000 | 1      | 437/404015360    | 23.2       |
| 1110100000000000 | 1110100010000000 | 1      | 19/92346368      | 4.4        |
| 0000000000000000 | 1100000000000000 | 2      | 13/215360        | 23.2       |

■ **Table 3** The Had<sub>4</sub>-to-2Lin(2) gadget  $G$  with minimal completeness among those that minimise  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ . The completeness of  $G$  is  $c(G) = 590174949/639271832$  and the infinity relaxed soundness is  $\text{rs}_\infty(G) = 141533171/159817958$ . The right most column tells how much of the total capacity is contained in each edge orbit. This column sums up to 100%.

| $f_1$            | $f_2$            | length | $G(f_1, f_2)$        | % of total |
|------------------|------------------|--------|----------------------|------------|
| 1100000000000000 | 1110000000000000 | 1      | 4899/799089790       | 33.0       |
| 1110000000000000 | 1111000000000000 | 1      | 11843/799089790      | 26.5       |
| 1110000000000000 | 1110100000000000 | 1      | 1427/1917815496      | 16.0       |
| 1110100000000000 | 1110100010000000 | 1      | 1427/19178154960     | 1.60       |
| 0000000000000000 | 1100000000000000 | 2      | 6094929/102283493120 | 22.9       |

### 4.3 Notable gadgets

There are two gadgets that are of particular interest. These are the gadgets with minimal completeness among those that maximises either  $\frac{1-\text{rs}(G)}{1-c(G)}$  or  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ . These gadgets are marked by crosses in Figure 7. The gadget with minimal completeness that maximises  $\frac{1-\text{rs}(G)}{1-c(G)}$  can be found in Table 2. The gadget with minimal completeness that maximises  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$  can be found in Table 3, and is also marked by a cross on the curve  $s(c)$  in Figures 1-3. The method used to construct such minimal completeness gadgets is slightly different compared to the construction of gadgets with fixed completeness. Propositions 2.23 and B.1 guarantees that gadgets with completeness  $1 - 2^{-k}$  can be used to maximise  $\frac{1-\text{rs}(G)}{1-c(G)}$  and  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ . This means that the maximum values of  $\frac{1-\text{rs}(G)}{1-c(G)}$  and  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$  can be computed by fixing the completeness to  $c(G) = 1 - 2^{-k}$ . Using these maximums, it is possible to slightly modify the objective of the LP such that its solution is the gadget with minimal completeness that maximises either  $\frac{1-\text{rs}(G)}{1-c(G)}$  or  $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ .

## 5 Conclusions

In this work, we have introduced a procedure called lifting for taking a Had <sub>$k$</sub> -to-2Lin(2) gadget for a fixed  $k$  and using that gadget to construct better and better Had <sub>$k'$</sub> -to-2Lin(2) gadgets, as  $k'$  tends to infinity. In order to be able to analyse this, both numerically and analytically, we made use of a relaxation of the (true) soundness, first introduced by Wiman [16] in their analysis of the Had<sub>4</sub>-to-2Lin(2) gadget. This procedure allowed us to show new inapproximability results of Max-2Lin(2), and most notably using  $k = 4$ , we have shown that Min-2Lin(2)-deletion has an inapproximability factor of  $\frac{73139148}{49096883} \approx 1.48969$ .

Some open problems still remain. The most obvious one is that it is likely within reach to carry out the analysis we did for  $k = 4$  also for  $k = 5$ . The main bottleneck is to find or write a very efficient LP solver that is able to handle large instances and give consistent and stable results. The solvers available to us were not quite able to get trustworthy results. This being said, without substantial new ideas we do not see how to attack the  $k = 6$  case.

Another open problem is to understand the best possible gadget reduction from  $\text{Had}_k$ -to- $2\text{Lin}(2)$  as  $k \rightarrow \infty$ . More specifically, which is the best possible inapproximability factor of  $\text{Min-}2\text{Lin}(2)$ -deletion attainable using such a gadget reduction? We were able to show an inapproximability factor of  $\frac{73139148}{49096883} \approx 1.48969$  using relaxed soundness. We have also shown that by using relaxed soundness, it is impossible to go above 2 (see Proposition B.1). Furthermore, it is known from a previous work [10, Theorem 6.1] that by using (non-relaxed) soundness,  $\frac{1}{1-e^{-0.5}} \approx 2.54$  is an upper bound. This leaves us with a fairly large gap. So it would be of interest to close this gap.

In comparison, by assuming the Unique Games Conjecture (UGC), it is possible to show that the inapproximability factor of  $\text{Min-}2\text{Lin}(2)$ -deletion can be made arbitrarily large. The main open problem here is to show this without assuming UGC. This however, is not possible to do using a gadget reduction from  $\text{Had}_k$ -to- $2\text{Lin}(2)$ , and would instead require a completely new approach.

Finally, as a concluding remark, it would be interesting to see if our ideas of lifting small gadgets and analysing them using a relaxed version of the (true) soundness, could be used in other applications. Maybe there are other gadgets out there that could be improved using a similar procedure?

---

## References

- 1 David Applegate, William Cook, Sanjeeb Dash, and Daniel Espinoza. Qsopt\_ex rational lp solver. <https://www.math.uwaterloo.ca/~bico/qsopt/ex/>. Accessed: 2023-09-20.
- 2 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, volume 62, pages 563–572, October 2010. doi:10.1109/FOCS.2010.59.
- 3 Per Austrin and Johan Håstad. On the usefulness of predicates. In *2012 IEEE 27th Conference on Computational Complexity*, pages 53–63, 2012. doi:10.1109/CCC.2012.18.
- 4 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *CoRR*, abs/0802.2300, 2008. arXiv:0802.2300.
- 5 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.
- 6 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 447–456, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488665.
- 7 Uriel Feige and Michael Langberg. The rpr2 rounding technique for semidefinite programs. *Journal of Algorithms*, 60(1):1–23, 2006. doi:10.1016/j.jalgor.2004.11.003.
- 8 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.
- 9 Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 1–10, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258536.

- 10 Johan Håstad, Sangxia Huang, Rajsekar Manokaran, Ryan O’Donnell, and John Wright. Improved NP-Inapproximability for 2-Variable Linear Equations. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 341–360, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.341.
- 11 S. Khot, G. Kindler, E. Mossel, and R. O’Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–154, 2004. doi:10.1109/FOCS.2004.49.
- 12 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, STOC ’02*, pages 767–775, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.510017.
- 13 Subhash Khot, 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, October 2018. doi:10.1109/FOCS.2018.00062.
- 14 Ryan O’Donnell and Yi Wu. An optimal sdp algorithm for max-cut, and equally optimal long code tests. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC ’08*, pages 335–344, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374425.
- 15 L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 617–626, 1996. doi:10.1109/SFCS.1996.548521.
- 16 Mårten Wiman. Improved inapproximability of max-cut through min-cut. Master’s thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

## A Max-Flow and symmetries

This section introduces the concepts of feasible flows and leaky flows, and show how to make use symmetries in a graph to more efficiently solve the Max-Flow problem. These Max-Flow techniques and concepts are used during the construction of gadgets. These techniques are very general, and become easier to explain without involving the intricacies of gadgets. Let us start by defining the Max-Flow problem as an LP.

► **Definition A.1.** A flow graph is a tuple  $G = (V, C, S, T)$ , where  $C(u, v) = C(v, u) \geq 0$  is the capacity of edge  $(u, v) \in V \times V$ , and  $S \subset V$  is a set of sources and  $T \subset V$  is a set of sinks, and  $S \cap T = \emptyset$ .

► **Definition A.2.** A flow  $w$  of a flow graph  $G = (V, C, S, T)$  is a function  $V \times V \rightarrow \mathbb{R}_{\geq 0}$ . The flow  $w$  is said to be feasible if and only if

$$w(v, u) + w(u, v) \leq C(u, v) \quad \forall v, u \in V, \tag{4}$$

$$\text{out}_w(v) = \text{in}_w(v) \quad \forall v \in V \setminus (S \cup T). \tag{5}$$

where

$$\text{out}_w(v) = \sum_{u \in V} w(v, u),$$

$$\text{in}_w(v) = \sum_{u \in V} w(u, v).$$

The value of a flow is defined as

$$\text{val}(w) = \sum_{s \in S} \text{out}_w(s) - \text{in}_w(s).$$

The value of the maximum flow of a flow graph  $G$  is denoted by  $\max\_flow(G)$ .

### A.1 Feasible flows and leaky flows

When solving a Max-Flow problem we normally require the flow to be conserved (constraint (5) above), meaning that the incoming flow into a node is equal to the outgoing flow. This is the definition of a feasible flow. However, to find an approximate solution to a Max-Flow problem, it can be helpful to relax the conservation of flows constraint, allowing for “leaks”. A flow that does not fulfil the conservation of flow constraint is called a *leaky flow*. This section aims to analyse the relation between leaky flows and feasible flows, with the goal of showing that if the leaks of a leaky flow are small, then there is a feasible flow with almost the same value as the leaky flow.

► **Definition A.3.** A flow  $\tilde{w}$  is said to be a leaky flow if constraint (4) is satisfied. The (signed) leak at node  $v$  be defined as  $\text{leak}_{\tilde{w}}(v) = \text{in}_{\tilde{w}}(v) - \text{out}_{\tilde{w}}(v)$  for  $v \in V \setminus (S \cup T)$ .

► **Remark A.4.** Note that a leaky flow  $\tilde{w}$  is also a feasible flow if and only if  $\text{leak}_{\tilde{w}}(v) = 0$  for all  $v \in V \setminus (S \cup T)$ .

The following theorem tells us that if the sum of absolute values of the leaks are small, then there is a feasible flow having almost the same value as the leaky flow. The implications from this is that we can use leaky flows to get an approximation of the true Max-Flow.

► **Theorem A.5.** Given a leaky flow  $\tilde{w}$  of a flow graph  $G = (V, C, S, T)$ , there exists a feasible flow  $w$  of  $G$  such that

$$\text{val}(w) \geq \text{val}(\tilde{w}) - \sum_{v \in V \setminus (S \cup T)} |\text{leak}_{\tilde{w}}(v)|.$$

**Proof.** Create a new graph  $\tilde{G} = (V \cup \{\tilde{s}, \tilde{t}\}, \tilde{C}, S \cup \{\tilde{s}\}, T \cup \{\tilde{t}\})$  with an additional new source node  $\tilde{s}$  and sink node  $\tilde{t}$ . We construct  $\tilde{C}$  using  $C$ . Firstly let  $\tilde{C}(u, v) = C(u, v)$  for all nodes  $u, v \in V$ . Secondly, for every  $v \in V \setminus (S \cup T)$  such that  $\text{leak}_{\tilde{w}}(v) > 0$ , let  $\tilde{C}(u, \tilde{t}) = \text{leak}_{\tilde{w}}(v)$ , and for every  $v \in V \setminus (S \cup T)$  such that  $\text{leak}_{\tilde{w}}(v) < 0$  let  $\tilde{C}(u, \tilde{s}) = -\text{leak}_{\tilde{w}}(v)$ . Finally let  $\tilde{C}$  be 0 in all other cases.

Note that for this new graph  $\tilde{G}$ , the leaky flow  $\tilde{w}$  can be extended into a feasible flow since all of the leaks can be routed to either  $\tilde{s}$  or  $\tilde{t}$  depending on the sign of the leakage. Furthermore, if we can show that

$$\max\_flow(\tilde{G}) \leq \max\_flow(G) + \sum_{v \in V \setminus (S \cup T)} |\text{leak}_{\tilde{w}}(v)|, \tag{6}$$

then that would imply the the Theorem.

To show (6) we use the Max-Flow Min-Cut Theorem. Note that any  $S$ - $T$  cut in  $\tilde{G}$  has a corresponding  $S$ - $T$  cut in  $G$  and vice versa since  $G$  and  $\tilde{G}$  share the same non-source/sink nodes. Additionally, note that the value of a  $S$ - $T$  cut in  $\tilde{G}$  can be bounded from above by the value of the corresponding cut in  $G$  plus the extra capacities in  $\tilde{G}$ . The conclusion from this is that

$$\begin{aligned} \max\_flow(\tilde{G}) &= \min\_cut(\tilde{G}) \\ &\leq \min\_cut(G) + \sum_{v \in V \setminus (S \cup T)} |\text{leak}_{\tilde{w}}(v)| \\ &= \max\_flow(G) + \sum_{v \in V \setminus (S \cup T)} |\text{leak}_{\tilde{w}}(v)|. \end{aligned} \quad \blacktriangleleft$$

## A.2 Symmetries of Max-Flow graphs

If a flow graph  $G = (V, C, S, T)$  has some kind of symmetry, then we can use them to more efficiently solve the Max-Flow problem. In our setting, the symmetries are described by a group  $H$  acting on  $V$  with the property that the capacities are invariant under the group action, meaning  $C(u, v) = C(h \cdot u, h \cdot v)$  for all  $h \in H$  and  $u, v \in V$ . Here  $h \cdot u$  denotes the group action of  $h$  on  $u$ .

► **Definition A.6.** *Given a flow graph  $G = (V, C, S, T)$  and a group  $H$  acting on  $V$ , then  $H$  is said to be a symmetry group of  $G$  if and only if  $\forall h \in H$ :*

1.  $h \cdot s \in S \forall s \in S$ ,
2.  $h \cdot t \in T \forall t \in T$ ,
3.  $C(u, v) = C(h \cdot u, h \cdot v) \forall h \in H$  and  $\forall u, v \in V$ .

Using  $G$  and the group  $H$  acting on  $V$ , we can create a new flow graph where the set of vertices is the quotient space  $V/H$ . This “compresses” the graph  $G$  into one vertex per orbit. Let the capacities between two orbits  $A, B \in V/H$  be the sum capacities over all pairs in  $A \times B$ .

► **Definition A.7.** *Given a flow graph  $G = (V, C, S, T)$  and a symmetry group  $H$  of  $G$ . Let the quotient flow graph  $G/H = (V/H, C/H, S/H, T/H)$  where  $V/H$  is the set of all orbits of  $V$  under the action of  $H$ , and similarly  $S/H$  is the set of orbits of  $S$  and  $T/H$  is the set of orbits of  $T$ . Let  $C/H$  be defined as a function  $V/H \times V/H \rightarrow \mathbb{R}$  such that*

$$(C/H)(A, B) = \sum_{u \in A} \sum_{v \in B} C(u, v)$$

for all  $A, B \in V/H$ .

What remains to show is that the original graph  $G$  and the compressed graph  $G/H$  has the same Max-Flow.

► **Theorem A.8.** *Given a flow graph  $G = (V, C, S, T)$  and a symmetry group  $H$  of  $G$ . Then  $\max\_flow(G) = \max\_flow(G/H)$ .*

**Proof.** First let us show that  $\max\_flow(G) \leq \max\_flow(G/H)$ . Let  $w$  be the max-flow of  $G$ . Now define  $w/H$  as a function from  $V/H \times V/H \rightarrow \mathbb{R}$  such that

$$(w/H)(A, B) = \sum_{a \in A} \sum_{b \in B} w(a, b).$$

What remains to show is that that  $w/H$  is a feasible flow of  $G/H$  and that  $\text{val}(w) = \text{val}(w/H)$  since those two properties would imply that  $\max\_flow(G) \leq \max\_flow(G/H)$ . Firstly, note that  $w/H$  fulfills (4) and (5) from Definition A.2 for the graph  $G/H$  since the constraints are linear. For example take constraint (4),

$$\begin{aligned} (w/H)(A, B) + (w/H)(B, A) &= \sum_{a \in A} \sum_{b \in B} w(a, b) + w(b, a) \\ &\leq \sum_{a \in A} \sum_{b \in B} C(a, b) \\ &= (C/H)(A, B). \end{aligned}$$



So  $w/H$  is a feasible flow of  $G/H$ . Secondly note that the value of  $w$  is the same as the value of  $w/H$  since

$$\begin{aligned} \text{val}(w/H) &= \sum_{A \in S/H} \text{out}_{w/H}(A) - \text{in}_{w/H}(A) \\ &= \sum_{A \in S/H} \sum_{s \in A} \text{out}_w(s) - \text{in}_w(s) \\ &= \sum_{s \in S} \text{out}_w(s) - \text{in}_w(s) \\ &= \text{val}(w). \end{aligned}$$

It remains to show that  $\max\_flow(G) \geq \max\_flow(G/H)$ . Let  $w'$  be a max-flow of  $G/H$ . Now define  $w : V \times V \rightarrow \mathbb{R}$  such that

$$w(a, b) = w'(H \cdot a, H \cdot b) \frac{C(a, b)}{(C/H)(H \cdot a, H \cdot b)}$$

where  $a, b \in V$  and  $H \cdot a$  is the orbit of  $a$  and  $H \cdot b$  is the orbit of  $b$ . What remains to show is that  $w(a, b)$  is a feasible flow of  $G$  and that the value of  $w$  is the same as the value of  $w'$ . Firstly, note that  $w/H$  fulfill constraints (4) and (5) from Definition A.2 for the graph  $G$  since the constraints are linear. For example take constraint (4),

$$\begin{aligned} w(a, b) + w(b, a) &= (w'(H \cdot a, H \cdot b) + w'(H \cdot b, H \cdot a)) \frac{C(a, b)}{(C/H)(H \cdot a, H \cdot b)} \\ &\leq (C/H)(H \cdot a, H \cdot b) \frac{C(a, b)}{(C/H)(H \cdot a, H \cdot b)} \\ &= C(a, b). \end{aligned}$$

Secondly note that the value of  $w$  is the same as  $w'$  since

$$\begin{aligned} \text{val}(w') &= \sum_{A \in S/H} \text{out}_{w'}(A) - \text{in}_{w'}(A) \\ &= \sum_{A \in S/H} \sum_{B \in V/H} w'(A, B) - w'(B, A) \\ &= \sum_{A \in S/H} \sum_{B \in V/H} (w'(A, B) - w'(B, A)) \left( \sum_{a \in A} \sum_{b \in B} \frac{C(a, b)}{(C/H)(A, B)} \right) \\ &= \sum_{A \in S/H} \sum_{B \in V/H} \sum_{a \in A} \sum_{b \in B} (w'(A, B) - w'(B, A)) \frac{C(a, b)}{(C/H)(A, B)} \\ &= \sum_{A \in S/H} \sum_{B \in V/H} \sum_{a \in A} \sum_{b \in B} w(a, b) - w(b, a) \\ &= \sum_{a \in S} \sum_{b \in V} w(a, b) - w(b, a) \\ &= \sum_{a \in S} \text{out}_w(a) - \text{in}_w(a) \\ &= \text{val}(w). \end{aligned}$$

So  $w$  is a feasible flow of  $G$  and  $\text{val}(w) = \text{val}(w')$ , so  $\max\_flow(G) \geq \max\_flow(G/H)$ . ◀

## B Properties of relaxed soundness

The relaxed soundness share many similarities with the (true) soundness. One example is the following Proposition, which is an analogue to Proposition 2.23 but for relaxed soundness.

► **Proposition B.1.** *For any  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}_k, \mathbb{Y}_k)$*

(a)

$$s(G) \leq \text{rs}(G).$$

(b) *There exists a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$  with completeness  $1 - 2^{-k}$  such that*

$$\frac{1 - \text{rs}(G)}{1 - c(G)} \leq \frac{1 - \text{rs}(\tilde{G})}{1 - c(\tilde{G})},$$

(c) *and for any  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$  with completeness  $1 - 2^{-k}$*

$$\frac{1 - \text{rs}(\tilde{G})}{1 - c(\tilde{G})} \leq 2.$$

**Proof.**

(a) Note that interpreting  $x_1$  and  $x_{-1}$  as being primary variables do not affect soundness, i.e.

$$s(G) = \max_{P \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \{x_1, x_{-1}\})} \mathbb{E}_{A \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \mathbb{Y}_k), A(x) = P(x), x \in \mathbb{X}_k \cup \{x_1, x_{-1}\}} \text{val}(A, G). \quad (7)$$

The reason for this is that there exists a degree of freedom in the choice of  $A$  since for any  $A$ ,  $\text{val}(A, G) = \text{val}(1 + A, G)$ . This means for example that we can add one extra constraint like  $A(x_1) = 1 + A(x_{-1}) = 1$  to the definition of  $s(G)$  without affecting its value.

Comparing (7) and the definition of relaxed soundness, we can clearly see that  $s(G) \leq \text{rs}(G)$  since the relaxed soundness is a less constrained maximisation problem compared to the right hand side of (7).

(b) This proof is analogous to the proof of [10, Proposition 2.29]. Note that by definition  $1 - c(\tilde{G})$  is the average length of edges  $(f_1, f_2)$  of the gadget  $\tilde{G}$ , weighted by  $\tilde{G}(f_1, f_2)$ . For  $\tilde{G}$  to have completeness  $1 - 2^{-k}$ , the edges in  $\tilde{G}$  need to have an average length of  $2^{-k}$ . Since there are no edges shorter than  $2^{-k}$ ,  $\tilde{G}$  can only put non-zero capacity on edges of length exactly  $2^{-k}$ .

Construct  $\tilde{G}$  using the following procedure. Start with  $G$ . Split up each edge  $(f_1, f_2)$  in  $G$  into an arbitrary path starting at  $f_1$ , ending at  $f_2$ , with edges of length  $2^{-k}$ , where the sum of lengths of edges in the path should be equal to the length of the original edge  $(f_1, f_2)$ . Remove the capacity of edge  $(f_1, f_2)$  and give each edge in the path the same capacity as the capacity of the original edge  $(f_1, f_2)$ . This will increase the total capacity of the graph by a factor of  $(1 - c(G))/2^k$ . As a final step, normalize the capacity by dividing the capacity of all edges by  $(1 - c(G))/2^k$ . Let the resulting graph be  $\tilde{G}$ . Note that  $\tilde{G}$  is a  $\text{Had}_k\text{-to-2Lin}(2)$  consisting only of edges of length  $2^{-k}$ , so its completeness is  $1 - 2^{-k}$ .

Recall that  $1 - \text{rs}(G)$  can be interpreted as the expected value of a Max-Flow problem on a fully connected  $2^k$ -dimensional hypercube, where the placements of sources and sinks have been randomised. Note that any feasible flow  $\omega$  of  $G$ , when scaled down by a factor of  $(1 - c(G))/2^{-k}$ , corresponds to a feasible flow of  $\tilde{G}$ . This implies that  $(1 - \text{rs}(G)) \leq (1 - \text{rs}(\tilde{G}))(1 - c(G))/2^{-k}$ .

The conclusion from this is that

$$\frac{1 - \text{rs}(\tilde{G})}{1 - c(\tilde{G})} = \frac{1 - \text{rs}(\tilde{G})}{2^{-k}} \geq \frac{1 - \text{rs}(G)}{1 - c(G)}.$$

- (c) Let  $\tilde{G}$  be the gadget from b). Recall that  $1 - \text{rs}(\tilde{G})$  can be interpreted as the expected value of a Max-Flow problem on a fully connected  $2^k$ -dimensional hypercube, where the placements of sources and sinks have been randomised. The capacities of this flow graph sum to 1.

Note that the sources and sinks correspond to affine functions, which have a normalised Hamming distance of at least  $1/2$ . Furthermore, since all edges in  $\tilde{G}$  has length  $2^{-k}$ , any path in  $\tilde{G}$  between a source and a sink must contain at least  $2^{k-1}$  edges.

For any flow graph, if all paths between sources and sinks contain at least  $2^{k-1}$  edges, and the sum of capacity over all edges in the graph is 1, then the maximum flow is at most  $2^{1-k}$ . So  $1 - \text{rs}(\tilde{G}) \leq 2^{1-k}$ , which implies that

$$\frac{1 - \text{rs}(\tilde{G})}{1 - c(\tilde{G})} \leq \frac{2^{1-k}}{2^{-k}} = 2. \quad \blacktriangleleft$$

► Remark B.2. Since the relaxed soundness is an upper bound of the true soundness, it follows that the NP-hardness result of Max-2Lin(2) as stated in Proposition 2.22 also holds for  $s = \text{rs}(G)$ .

## C Affine maps and lifts

Recall that the  $\text{rsLP}(G)$  can be interpreted as the expected value of a Max-Flow problem with a randomised source/sink placement over a fully connected  $2^k$ -dimensional hypercube, where the nodes are indexed by Boolean functions  $f \in \mathcal{F}_k$ . The source/sink nodes are indexed by affine Boolean functions. In order to be able to describe the symmetries of these graphs, we want to study mappings  $M : \mathcal{F}_k \rightarrow \mathcal{F}_k$  with the following properties:

1. Source and sink nodes map to source and sink nodes, i.e. if  $f$  is an affine function then  $M(f)$  is also an affine function.
2. The length of all edges  $\{v_{f_1}, v_{f_2}\}$  are preserved by the mapping, i.e.  $\text{dist}(M(f_1), M(f_2)) = \text{dist}(f_1, f_2)$ .

There is a natural choice of mappings from  $\mathcal{F}_k \rightarrow \mathcal{F}_k$  for which Property 1 and 2 hold. Additionally as a bonus, the same natural choice of mappings can also be extended to construct mappings from  $\mathcal{F}_k \rightarrow \mathcal{F}_{k'}$ ,  $k \leq k'$ , and still have that both Property 1 and 2 hold. This can then be used to embed the  $2^k$ -dimensional hypercube in the  $2^{k'}$ -dimensional hypercube.

► Definition C.1. Let  $M_{A,b,\beta,c} : \mathcal{F}_k \rightarrow \mathcal{F}_{k'}$  be defined as

$$M_{A,b,\beta,c}(f)(y) = f(Ay + b)(-1)^c \chi_\beta(y),$$

where  $k, k' \in \mathbb{Z}_{>0}$ ,  $k \leq k'$ ,  $y \in \mathbb{F}_2^{k'}$ ,  $A \in \mathbb{F}_2^{k \times k'}$  is a full rank matrix,  $b \in \mathbb{F}_2^k$ ,  $c \in \mathbb{F}_2$  and  $\beta \in \mathbb{F}_2^{k'}$ . Let  $\mathcal{M}_{k \rightarrow k'}$  denote the set of all maps  $M_{A,b,\beta,c}$  from  $\mathcal{F}_k \rightarrow \mathcal{F}_{k'}$ . For convenience, we often denote  $M_{A,b,\beta,c}$  by  $M$ , where the  $A, b, \beta, c$  are all implicit.

## 11:28 On the NP-Hardness Approximation Curve for Max-2Lin(2)

Since these mappings are reminiscent of affine maps from linear algebra, we call them *affine maps*. However, they are not affine maps in the classical sense.

The function  $M(f) \in \mathcal{F}_{k'}$  is called the *M-lift of f*. It is not hard to see that the *M-lift* of an affine function is an affine function. More generally, *M-lifts* always preserve the dimension of Boolean functions.

► **Proposition C.2.** *Given  $f \in \mathcal{F}_k$  and  $M \in \mathcal{M}_{k \rightarrow k'}$ ,  $k \leq k'$ , then  $\dim(M(f)) = \dim(f)$ .*

**Proof.** It follows from a direct calculation that

$$M_{A,b,\beta,c}(f)(y) = (-1)^c \sum_{\alpha \in \{0,1\}^k} \chi_{A^T \alpha + \beta}(y) \hat{f}_\alpha \chi_b(\alpha).$$

This shows that the affine mapping  $M$  moves  $\text{affine}(f)$  to  $\text{affine}(M(f)) = \{A^T \alpha + \beta : \alpha \in \text{affine}(f)\}$ . Furthermore, since  $A$  is a full rank matrix,  $\dim(f) = \dim(M(f))$ . ◀

Affine maps also preserve the (normalised Hamming) distance of affine functions.

► **Proposition C.3.** *Given  $f_1, f_2 \in \mathcal{F}_k$  and  $M \in \mathcal{M}_{k \rightarrow k'}$ ,  $k \leq k'$ , then  $\text{dist}(M(f_1), M(f_2)) = \text{dist}(f_1, f_2)$ .*

**Proof.** Let  $M = M_{A,b,\beta,c}$ . Note that  $\text{dist}(M(f_1), M(f_2))$  only depends on  $A$  and  $b$  since

$$\begin{aligned} \text{dist}(M(f_1), M(f_2)) &= \frac{1}{2^{k'}} \sum_{y \in \mathbb{F}_2^{k'}} \frac{1 - M(f_1)(y)M(f_2)(y)}{2} \\ &= \frac{1}{2^{k'}} \sum_{y \in \mathbb{F}_2^{k'}} \frac{1 - f_1(Ay + b)f_2(Ay + b)}{2}. \end{aligned}$$

Furthermore, since  $A$  is a full rank  $k \times k'$  Boolean matrix, the kernel of  $A$  has dimension  $k' - k$  and size  $2^{k'-k}$ , so

$$\sum_{y \in \{0,1\}^{k'}} f_1(Ay + b)f_2(Ay + b) = 2^{k'-k} \sum_{x \in \{0,1\}^k} f_1(x)f_2(x).$$

This shows that  $\text{dist}(M(f_1), M(f_2)) = \text{dist}(f_1, f_2)$ . ◀

The last notable property of the affine maps is that they form a group under composition. This property is needed to be able to apply the techniques from Appendix A.2 to the  $\text{rsLP}(G)$  and to the  $\text{rs}_\infty \text{LP}(G)$  in order to “compress” them.

► **Proposition C.4.**  *$\mathcal{M}_{k \rightarrow k}$  under composition forms a group.*

**Proof.** The composition of two affine maps  $M_{A',b',\beta',c'} \circ M_{A,b,\beta,c}$ , is an affine map  $M_{A'',b'',\beta'',c''}$ , where

$$\begin{aligned} A'' &= AA', \\ b'' &= Ab' + b, \\ \beta'' &= (A')^T \beta + \beta', \\ c'' &= (b', \beta) + c' + c. \end{aligned}$$

Furthermore, the left and right inverse of an affine map  $M_{A,b,\beta,c}$  is given by  $M_{A',b',\beta',c'}$  where

$$\begin{aligned} A' &= A^{-1}, \\ b' &= A^{-1}b, \\ \beta' &= (A^{-1})^T \beta, \\ c' &= c + (A^{-1}b, \beta). \end{aligned}$$

This shows that  $\mathcal{M}_{k \rightarrow k}$  forms a group under composition. ◀

### C.1 $M$ -lifts of sink and sources

Recall that the source/sink placements of the  $\text{rsLP}(G)$  and the  $\text{rs}_\infty\text{LP}(G)$  are described using a Boolean function  $g \in \mathcal{F}_k$ ,

$$g(\alpha) = \begin{cases} 1 & \text{iff } v_{\chi_\alpha} \text{ is a sink,} \\ -1 & \text{iff } v_{\chi_\alpha} \text{ is a source.} \end{cases}$$

Note that  $M$ -lifts move the sink and source nodes. If  $k = k'$ , then the  $M$ -lift permutes the sink and source nodes. If  $k < k'$ , then the  $M$ -lift “lifts” the sink and source nodes onto a higher dimensional hypercube. This means that there exists multiple different source/sink placements  $g' \in \mathcal{F}_{k'}$  that all match the lifted positions of the sinks and sources. The condition for when an  $M$ -lift of a source/sink placement  $g \in \mathcal{F}_k$  is described by a source/sink placement  $g' \in \mathcal{F}_{k'}$  is given by the following proposition.

► **Proposition C.5.** *An  $M$ -lift will map sink nodes in  $\mathcal{F}_k$  onto sink nodes of  $\mathcal{F}_{k'}$  and source nodes in  $\mathcal{F}_k$  onto source nodes in  $\mathcal{F}_{k'}$  if and only if*

$$M_{A^T, \beta, b, c}(g') = g.$$

**Proof.**

Note that the  $M_{A,b,\beta,c}$ -lift of  $\chi_\alpha$  is  $M_{A,b,\beta,c}(\chi_\alpha) = \chi_{A^T \alpha + \beta}(x)(-1)^c \chi_b(\alpha)$ . Using the source/sink placement  $g'$  we can tell whether a node  $v_{\chi_\alpha}$  is lifted onto a sink node or a source node,

$$g'(A^T \alpha + \beta)(-1)^c \chi_b(\alpha) = \begin{cases} 1 & \text{iff } v_{M_{A,b,\beta,c}(\chi_\alpha)} \text{ is a sink according to } g', \\ -1 & \text{iff } v_{M_{A,b,\beta,c}(\chi_\alpha)} \text{ is a source according to } g'. \end{cases}$$

This implies that the sufficient and necessary condition to make all sinks in  $\mathcal{F}_k$  to be  $M_{A,b,\beta,c}$ -lifted to sinks in  $\mathcal{F}_{k'}$  and all sources in  $\mathcal{F}_k$  to be  $M_{A,b,\beta,c}$ -lifted to sources in  $\mathcal{F}_{k'}$ , is that

$$g(\alpha) = g'(A^T \alpha + \beta)(-1)^c \chi_b(\alpha)$$

for all  $\alpha \in \mathbb{F}_2^k$ . This is identical to requiring that  $M_{A^T, \beta, b, c}(g') = g$ . ◀

► **Definition C.6.** *The operator  $M_{A^T, \beta, b, c}$  is denoted by  $M_{A,b,\beta,c}^\#$ .*

### C.2 Lifting gadgets and flows

It is possible to extend the definition of  $M$ -lifting to  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets  $G$  by defining  $M \cdot G$  as

$$(M \cdot G)(f'_1, f'_2) = \sum_{\substack{f_1 \in M^{-1}(f'_1), \\ f_2 \in M^{-1}(f'_2)}} G(f_1, f_2).$$

## 11:30 On the NP-Hardness Approximation Curve for Max-2Lin(2)

This moves the capacity  $G(f_1, f_2)$  of edge  $\{v_{f_1}, v_{f_2}\}$  onto edge  $\{v_{M(f_1)}, v_{M(f_2)}\}$ . Furthermore, let the *full  $k \rightarrow k'$  lift* of  $G$  be defined as the average of all possible  $M$ -lifts, i.e.

$$\text{lift}_{k \rightarrow k'}(G) = \frac{1}{|\mathcal{M}_{k \rightarrow k'}|} \sum_{M \in \mathcal{M}_{k \rightarrow k'}} (M \cdot G).$$

Completely analogue to the definition of  $M$ -lifts of gadgets, let the  $M$ -lift of a flow  $w$  of the  $\text{rs}(G)$  LP be defined as

$$(M \cdot w)(f'_1, f'_2, g') = \sum_{\substack{f_1 \in M^{-1}(f'_1), \\ f_2 \in M^{-1}(f'_2)}} w(f_1, f_2, M^\#(g')),$$

and let the full  $k \rightarrow k'$  lift of  $w$  be defined as

$$\text{lift}_{k \rightarrow k'}(w) = \frac{1}{|\mathcal{M}_{k \rightarrow k'}|} \sum_{M \in \mathcal{M}_{k \rightarrow k'}} (M \cdot w).$$

By connecting these two concepts of lifting gadgets and flows, we can show the following proposition.

► **Proposition C.7.** *The full lift of  $G$  is a  $\text{Had}_k$ -to-2Lin(2) gadget  $G'$  where  $c(G') = c(G)$  and  $\text{rs}(G') \leq \text{rs}(G)$ .*

**Proof.** Let  $w$  be a feasible flow of  $G$  and let  $w' = \text{lift}_{k \rightarrow k'}(w)$ . Note that  $w'$  is a feasible flow of  $G'$  since the capacity of  $G$  is lifted together with the flow  $w$ . So constraints (1) and (2) are satisfied by  $w'$ . Additionally,

$$\mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w').$$

since any lift preserves the amount of flow going in and out of sink nodes and source nodes. ◀

The final Proposition that we need for Appendix D is that the full lift of a leaky flow  $w$  of the  $\text{rsLP}(G)$  is a leaky flow of the  $\text{rsLP}(G')$ , and that the full lift does not affect the value of the flow. This is a fundamental property of lifts that is used in Appendix D to upper bound  $\text{rs}(G')$  when  $k' \rightarrow \infty$ .

► **Proposition C.8.** *Let  $G'$  be the full lift of  $G$ , and let  $w'$  be the full lift of a leaky flow  $w$  of the  $\text{rsLP}(G)$ . Then  $w'$  is a leaky flow of the  $\text{rsLP}(G')$ , and  $\mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w')$ .*

**Proof.** Let  $w$  be a leaky flow of  $G$  and let  $w' = \text{lift}_{k \rightarrow k'}(w)$ . Note that constraint (1) is satisfied by  $w'$  since the capacity of  $G$  is lifted together with the flow  $w$ . So  $w'$  is a leaky flow. Additionally,

$$\mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w').$$

since any lift preserves the amount of flow going in and out of sink nodes and source nodes. ◀

## D Proving that $\text{rs}_\infty(G)$ can be attained in the limit

The goal of this section is to prove Lemma 3.11, which relates the infinity relaxed soundness to the relaxed soundness. Let  $G$  be the  $\text{Had}_k$ -to-2Lin(2) gadget in Lemma 3.11 and let  $w$  be the optimal flow of the  $\text{rs}_\infty \text{LP}(G)$ , which implies that  $\text{rs}_\infty(G) = 1 - \mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w)$ . Let  $k'$  be some integer greater than  $k$  and define  $G' = \text{lift}_{k \rightarrow k'}(G)$  and  $w' = \text{lift}_{k \rightarrow k'}(w)$ . According to Proposition C.7  $G'$  is a  $\text{Had}_k$ -to-2Lin(2) with  $c(G') = c(G)$  and according to Proposition C.8  $w'$  is a leaky flow of the  $\text{rsLP}(G')$  and  $\mathbb{E}_{g \in \mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w')$ . We prove that as  $k'$  tends to infinity the total leakage of  $G'$  converges to 0. After we have established this, Lemma 3.11 follows from Theorem A.5.

### D.1 Total leakage approaches 0 as $k' \rightarrow \infty$

Let us start by formally defining the leaks of  $w$  and  $w'$ , where  $w$  is a leaky flow of the rsLP( $G$ ) and  $w'$  is a leaky flow of the rsLP( $G'$ ). Recall that the rsLP( $G'$ ) describe the expectation of the maximum flow of a graph with a random source/sink placement  $g' \in \mathcal{F}_{k'}$ . It is for this reason that the total leakage of  $w'$  is defined as an expectation over  $g' \in \mathcal{F}_{k'}$  of the total leakage of the graph with source/sink placement given by  $g'$ .

► **Definition D.1.** Let  $L_{k'}$  denote the total leakage of  $w'$ ,

$$L_{k'} = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \sum_{\substack{f' \in \mathcal{F}_{k'} \\ \text{s.t. } \dim(f') > 0}} |\text{leak}_{w'}(f', g')| \right),$$

where

$$\begin{aligned} \text{leak}_{w'}(f', g') &= \text{out}_{w'}(f', g') - \text{in}_{w'}(f', g') \\ &= \frac{1}{|\mathcal{M}_{k \rightarrow k'}|} \sum_{M \in \mathcal{M}_{k \rightarrow k'}} \left( \sum_{\substack{f \in \mathcal{F}_k \\ \text{s.t. } M(f) = f'}} \text{leak}_w(f, M^\#(g')) \right). \end{aligned}$$

The aim of this subsection is to prove that  $L_{k'} \rightarrow 0$  as  $k' \rightarrow \infty$ . We do this by proving the following upper bound on  $L_{k'}$  through a second order moment analysis.

► **Proposition D.2.**

$$L_{k'} \leq \frac{2^{2k+k}}{\sqrt{2^{k'} - 2^k}}.$$

The proof of Proposition D.2 relies on the following Proposition describing the relationship between random pairs of affine maps  $M_1, M_2 \in \mathcal{M}_{k \rightarrow k'}$  such that  $M_1(f) = M_2(f)$  for some fixed  $f \in \mathcal{F}_k$ .

► **Definition D.3.** Given  $M_{A,b,\beta,c} \in \mathcal{M}_{k \rightarrow k'}$ , let  $T_M : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^{k'}$  denote the affine map  $T_M(x) = Ax + \beta$ . Furthermore, let  $\text{affine}(M_{A,b,\beta,c})$  denote the affine subspace  $\{T_M(x) : x \in \mathbb{F}_2^k\} \subseteq \mathbb{R}^{k'}$ .

► **Proposition D.4.** Given  $f \in \mathcal{F}_k$  and  $f' \in \mathcal{F}_{k'}$  with  $\dim(f) = \dim(f') = d$ . Then

$$|\{(M_1, M_2) \in \mathcal{N}_{f \rightarrow f'} \times \mathcal{N}_{f \rightarrow f'} : \dim(\text{affine}(M_1) \cap \text{affine}(M_2)) > d\}| \leq |\mathcal{N}_{f \rightarrow f'}|^2 \frac{(2^k - 2^d)^2}{2^{k'} - 2^d},$$

where  $\mathcal{N}_{f \rightarrow f'} = \{M \in \mathcal{M}_{k \rightarrow k'} : M(f) = f'\}$  denotes the set of affine maps in  $\mathcal{M}_{k \rightarrow k'}$  that lifts  $f$  to  $f'$ .

**Proof.** Note that for any  $M_1, M_2 \in \mathcal{N}_{f \rightarrow f'}$ , the dimension of  $\text{affine}(M_1) \cap \text{affine}(M_2)$  is at least  $d$ , since according to the proof of Proposition C.2 both  $T_{M_1}$  and  $T_{M_2}$  must map  $\text{affine}(f)$  onto  $\text{affine}(f')$ , so  $\dim(\text{affine}(M_1) \cap \text{affine}(M_2) \cap \text{affine}(f')) = d$ . However, the two maps  $T_{M_1}$  and  $T_{M_2}$  can map the complement of  $\text{affine}(f)$  in different ways since there is no restriction to how they map the complement of  $\text{affine}(f)$ .

## 11:32 On the NP-Hardness Approximation Curve for Max-2Lin(2)

Fix  $M_1$  and uniformly at random pick  $M_2$  from  $\mathcal{N}_{f \rightarrow f'}$ . Given any fix  $x \notin \text{affine}(f)$ , the probability that  $T_{M_2}(x) \in \text{affine}(M_1)$  is  $(2^k - 2^d)/(2^{k'} - 2^d)$  since  $|\text{affine}(M_1) \setminus \text{affine}(f')| = 2^k - 2^d$  and  $T_{M_2}(x)$  is uniformly distributed over the complement of  $\text{affine}(f')$ . Taking a union bound over all  $x \notin \text{affine}(f)$  shows that

$$P_{M_2 \in \mathcal{N}_{f \rightarrow f'}}[\dim(\text{affine}(M_1) \cap \text{affine}(M_2)) > d] \leq \frac{(2^k - 2^d)^2}{2^{k'} - 2^d}.$$

Proposition D.4 follows directly from this inequality.  $\blacktriangleleft$

The takeaway from Proposition D.4 is that if  $M_1$  and  $M_2$  are two random affine maps such that  $M_1(f) = M_2(f)$  for some fixed  $f \in \mathcal{F}_k$ , then with high probability  $\text{affine}(M_1) \cap \text{affine}(M_2) = \text{affine}(f)$ . This allows us to create a bound on the second order moment of the terms that define  $L_{k'}$ .

**► Lemma D.5.** *Given  $f \in \mathcal{F}_k$ ,  $f' \in \mathcal{F}_{k'}$  and  $g' \in \mathcal{F}_{k'}$ , where  $\dim(f) = \dim(f') = d > 0$ , then*

$$\mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \left| \sum_{M \in \mathcal{N}_{f \rightarrow f'}} \text{leak}_w(f, M^\#(g')) \right|^2 \right) \leq |\mathcal{N}_{f \rightarrow f'}|^2 \frac{(2^k - 2^d)^2}{2^{k'} - 2^d}.$$

**Proof.** Expanding the square we need to prove that,

$$\sum_{M_1, M_2 \in \mathcal{N}_{f \rightarrow f'}} \mathbb{E}_{g' \in \mathcal{F}_{k'}} (\text{leak}_w(f, M_1^\#(g')) \text{leak}_w(f, M_2^\#(g'))) \leq |\mathcal{N}_{f \rightarrow f'}|^2 \frac{(2^k - 2^d)^2}{2^{k'} - 2^d}.$$

Split the terms up into two cases, either  $\dim(\text{affine}(M_1) \cap \text{affine}(M_2)) > d$  or  $\dim(\text{affine}(M_1) \cap \text{affine}(M_2)) = d$ . By Proposition D.4 the number of terms of the first type is at most  $|\mathcal{N}_{f \rightarrow f'}|^2 (2^k - 2^d)^2 / (2^{k'} - 2^d)$ . Each term is bounded by one since the sum of capacities in the rs( $G$ ) LP is equal to 1, so the absolute value of a leak is always smaller than or equal to 1 at any node and for any source/sink placement.

In the other case, when  $\dim(\text{affine}(M_1) \cap \text{affine}(M_2)) = d$ , then the two random functions  $M_1^\#(g')$  and  $M_2^\#(g')$  are equal on  $\text{affine}(f)$ , and independently uniformly random  $\{1, -1\}$  on the complement of  $\text{affine}(f)$ . This allows us to rewrite the expectation over  $g'$  as

$$\begin{aligned} & \mathbb{E}_{g' \in \mathcal{F}_{k'}} (\text{leak}_w(f, M_1^\#(g')) \text{leak}_w(f, M_2^\#(g'))) \\ &= \mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \text{leak}_w(f, M_1^\#(g')) \mathbb{E}_{\substack{g'_2 \in \mathcal{F}_{k'} \\ \text{s.t. } M_2^\#(g'_2)|_{\text{affine}(f)} = M_1^\#(g')|_{\text{affine}(f)}}} \text{leak}_w(f, M_2^\#(g'_2)) \right) \\ &= \mathbb{E}_{g \in \mathcal{F}_k} \left( \text{leak}_w(f, g) \mathbb{E}_{\substack{g_2 \in \mathcal{F}_k \\ \text{s.t. } g_2|_{\text{affine}(f)} = g|_{\text{affine}(f)}}} \text{leak}_w(f, g_2) \right). \end{aligned}$$

This is equal to 0, since for any infinity relaxed flow  $w$  (see Definition 3.7) the expectation of  $\text{leak}_w(f, g_2)$  over  $g_2$  given  $g$  is 0.  $\blacktriangleleft$

We are now at the point where we can prove Proposition D.2 using Lemma D.5.



**Proof of Proposition D.2.** A trivial upper bound of  $L_{k'}$  using the triangle inequality is

$$L_{k'} \leq \frac{1}{|\mathcal{M}_{k \rightarrow k'}|} \sum_{f \in \mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t. \dim(f') > 0}} \mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \left| \sum_{M \in \mathcal{N}_{f \rightarrow f'}} \text{leak}_w(f, M^\#(g')) \right| \right).$$

Applying Jensen's inequality to the expectation over  $g' \in \mathcal{F}_{k'}$  gives

$$\mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \left| \sum_{M \in \mathcal{N}_{f \rightarrow f'}} \text{leak}_w(f, M^\#(g')) \right| \right) \leq \sqrt{\mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \left| \sum_{M \in \mathcal{N}_{f \rightarrow f'}} \text{leak}_w(f, M^\#(g')) \right|^2 \right)},$$

which according to Lemma D.5 can be further upper bounded by

$$\begin{aligned} \sqrt{\mathbb{E}_{g' \in \mathcal{F}_{k'}} \left( \left| \sum_{M \in \mathcal{N}_{f \rightarrow f'}} \text{leak}_w(f, M^\#(g')) \right|^2 \right)} &\leq \frac{2^k - 2^{\dim(f)}}{\sqrt{2^{k'} - 2^{\dim(f)}}} |\mathcal{N}_{f \rightarrow f'}| \\ &\leq \frac{2^k}{\sqrt{2^{k'} - 2^k}} |\mathcal{N}_{f \rightarrow f'}|. \end{aligned}$$

We have so far shown that

$$L_{k'} \leq \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f \in \mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t. \dim(f') > 0}} \frac{|\mathcal{N}_{f \rightarrow f'}|}{|\mathcal{M}_{k \rightarrow k'}|}.$$

Finally, note that  $\sum_{f' \in \mathcal{F}_{k'}} |\mathcal{N}_{f \rightarrow f'}| = |\mathcal{M}_{k \rightarrow k'}|$  since  $\mathcal{N}_{f \rightarrow f'}$  are disjoint subsets of  $\mathcal{M}_{k \rightarrow k'}$  for different  $f' \in \mathcal{F}_{k'}$  and their union over  $f' \in \mathcal{F}_{k'}$  is equal to  $\mathcal{M}_{k \rightarrow k'}$ . So

$$L_{k'} \leq \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f \in \mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t. \dim(f') > 0}} \frac{|\mathcal{N}_{f \rightarrow f'}|}{|\mathcal{M}_{k \rightarrow k'}|} \leq \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f \in \mathcal{F}_k} 1 \leq \frac{2^{2k+k}}{\sqrt{2^{k'} - 2^k}}. \blacktriangleleft$$

## D.2 The proof of Lemma 3.11

All that remains is to tie up the loose ends by proving Lemma 3.11 using Proposition D.2 combined with Theorem A.5.

**Proof of Lemma 3.11.** Since  $w'$  is a leaky flow of the rsLP( $G'$ ), it follows from Theorem A.5 that there exists a feasible flow  $\tilde{w}'$  of the rsLP( $G'$ ) such that

$$\mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(\tilde{w}') + L_{k'} \geq \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w').$$

Note that  $\text{rs}(G') \geq 1 - \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(\tilde{w}')$  since  $\tilde{w}'$  is a feasible flow of the rsLP( $G'$ ). Furthermore, recall that  $\text{rs}_\infty(G) = 1 - \mathbb{E}_{g' \in \mathcal{F}_{k'}} \text{val}_{g'}(w')$ . So

$$\text{rs}(G') - L_{k'} \leq \text{rs}_\infty(G).$$

Proposition D.2 implies that  $L_{k'} \rightarrow 0$  as  $k' \rightarrow \infty$ , which proves that  $\forall \varepsilon > 0$  there exists a gadget  $G'$  with  $c(G') = c(G)$  such that  $\text{rs}(G') - \varepsilon \leq \text{rs}_\infty(G)$ .  $\blacktriangleleft$

## E Gadget construction and verification

This section contains the details for how to practically compute  $\text{Had}_k\text{-to-2Lin}(2)$  gadgets using the  $\text{rsLP}(G)$  and the  $\text{rs}_\infty\text{LP}(G)$ . These LPs have far too many variables and constraints to directly be solved by a computer when  $k \geq 4$ . The solution is to make use of the symmetries of the LP:s to construct smaller LP:s with the same optimum. This is done in two steps. Step 1 is to use Proposition C.7 to argue that best gadgets are the symmetrical gadgets. This means that we only need to take into account symmetrical gadgets when solving the  $\text{rsLP}(G)$  and the  $\text{rs}_\infty\text{LP}(G)$ . Step 2 is to use the fact that if  $G$  is symmetrical, then Theorem A.8 allows us to compress the LP, merging a huge number of variables into a single variable.

### E.1 Symmetrical $\text{Had}_k\text{-to-2Lin}(2)$ gadgets are optimal

The meaning of a  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}, \mathbb{Y})$  being *optimal* is that there exists no  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(\tilde{G}, \mathbb{X}, \mathbb{Y})$  such that  $c(G) = c(\tilde{G})$  and  $\text{rs}(G) > \text{rs}(\tilde{G})$ . The following Proposition states that symmetric gadgets are optimal. By symmetric, we refer to the property that the gadget  $G$  is invariant under  $M$ -lifts.

► **Proposition E.1.** *Given any  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(G, \mathbb{X}, \mathbb{Y})$ , there exists a symmetric  $\text{Had}_k\text{-to-2Lin}(2)$  gadget  $(\tilde{G}, \mathbb{X}, \mathbb{Y})$  such that  $c(G) = c(\tilde{G})$  and  $\text{rs}(G) \geq \text{rs}(\tilde{G})$ .*

**Proof.** Let  $\tilde{G} = \text{lift}_{k \rightarrow k}(G)$ . According to Proposition C.7,  $c(G) = c(\tilde{G})$  and  $\text{rs}(G) \geq \text{rs}(\tilde{G})$ . Furthermore,  $\tilde{G}$  is a symmetric gadget since for any  $f_1, f_2 \in \mathcal{F}_k$  and  $M \in \mathcal{M}_{k \rightarrow k}$ ,

$$\begin{aligned} (M \cdot \tilde{G})(f_1, f_2) &= \frac{1}{|\mathcal{M}_{k \rightarrow k}|} \sum_{M_2 \in \mathcal{M}_{k \rightarrow k}} ((M \circ M_2) \cdot \tilde{G})(f_1, f_2) \\ &= \frac{1}{|\mathcal{M}_{k \rightarrow k}|} \sum_{M_2 \in M \circ \mathcal{M}_{k \rightarrow k}} (M_2 \cdot \tilde{G})(f_1, f_2). \end{aligned}$$

According to Proposition C.4,  $\mathcal{M}_{k \rightarrow k}$  forms a group, so  $M \circ \mathcal{M}_{k \rightarrow k} = \mathcal{M}_{k \rightarrow k}$ . We have shown that  $M \cdot \tilde{G} = \tilde{G}$  and thus  $\tilde{G}$  is a symmetric gadget. ◀

### E.2 Compressing the $\text{rsLP}(G)$ and $\text{rs}_\infty\text{LP}(G)$

As discussed earlier, both the  $\text{rsLP}(G)$  and the  $\text{rs}_\infty\text{LP}(G)$  can be interpreted as Max-Flow problems. Furthermore, if  $G$  is symmetric under  $M$ -lifts, then  $\mathcal{M}_{k \rightarrow k}$  is a symmetry group for both of these Max-Flow problems. This means that we can apply Theorem A.8 to compress the Max-Flow problems, giving us the *compressed*  $\text{rsLP}(G)$  and the *compressed*  $\text{rs}_\infty\text{LP}(G)$ .

One of the symmetries that the compression is able to capture is that many different source/sink placements are equivalent. In a sense, the source/sink placements of the compressed LPs consist of one representative source/sink placement from each set of equivalent source/sink placements. This symmetry turns out to be the main contributor as to why the compressed LP is significantly smaller than the original LP.

Without the compression, the LPs each have  $2^{3 \cdot 2^k}$  variables, which for  $k \geq 4$  is computationally infeasible. However, even with the compression, for  $k = 4$  the LPs are still large enough that it is computationally challenging to solve them.

#### E.2.1 Further restricting the compressed LPs

To further restrict the size of the LPs in the case of  $k = 4$ , we heuristically identify a list of beneficial gadget variables by solving the compressed LPs with floating point numbers

■ **Table 4** Sizes of the  $\text{rsLP}(G)$  and  $\text{rs}_\infty\text{LP}(G)$  for  $\text{Had}_2\text{-to-2Lin}(2)$  gadgets  $G$ . The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints. All variables have the implicit constraint of being non-negative.

|            | $\text{rsLP}(G)$ |     |     | $\text{rs}_\infty\text{LP}(G)$ |     |     |
|------------|------------------|-----|-----|--------------------------------|-----|-----|
| Original   | 163              | 343 | 534 | 163                            | 343 | 534 |
| Compressed | 23               | 38  | 106 | 23                             | 38  | 106 |

■ **Table 5** Sizes of the  $\text{rsLP}(G)$  and  $\text{rs}_\infty\text{LP}(G)$  for  $\text{Had}_3\text{-to-2Lin}(2)$  gadgets  $G$ . The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints. All variables have the implicit constraint of being non-negative.

|            | $\text{rsLP}(G)$ |                |                | $\text{rs}_\infty\text{LP}(G)$ |                |                |
|------------|------------------|----------------|----------------|--------------------------------|----------------|----------------|
| Original   | $8 \cdot 10^6$   | $2 \cdot 10^7$ | $5 \cdot 10^7$ | $8 \cdot 10^6$                 | $2 \cdot 10^7$ | $5 \cdot 10^7$ |
| Compressed | 298              | 546            | 2330           | 243                            | 462            | 1987           |

using Gurobi. Any gadget variable that is given non-zero weight in at least one floating point solution is added to the list. Using this list, we define the *restricted compressed LP* as the compressed LP but with all other gadget variables that are not on the list, removed. The list we use can be found in Table 9 in Appendix F. Note that one possible drawback to restricting the LPs like this is that the restriction could lead to construction of sub-optimal gadgets.

Tables 4–6 show the sizes of the LPs depending on if compression or restriction is being applied. Note that the restricted and compressed LP:s have significantly fewer variables than the original LP:s.

There is a special case where we do not need the restrictions. If the completeness of a gadget is  $1 - 2^{-k}$ , then the gadget only has non-zero weight on edges of length  $2^{-k}$ . There are comparatively relatively few edges of length  $2^{-k}$ . This allows us to directly construct the gadget by solving the non-restricted LP. So in the case of completeness  $1 - 2^{-k}$ , the gadgets we construct are guaranteed to be optimal since we do not make use of any restrictions.

### E.3 Implementation details

The compressed  $\text{rsLP}(G)$  and compressed  $\text{rs}_\infty\text{LP}(G)$  are constructed using a Python script where all of the calculations are done using integer arithmetic. The script makes use of affine maps to efficiently compute the symmetries of the two LPs, in order to compress them. The time and memory complexities of the script are roughly  $O(2^{2 \cdot 2^k})$ , so the script is able to handle  $k = 2, 3$  and  $4$ . In theory it would be possible to also make the script support  $k = 5$ , but that would require both more powerful hardware, as well as improving the time complexity to roughly  $O(2^{2^k})$  time.

■ **Table 6** Sizes of the  $\text{rsLP}(G)$  and  $\text{rs}_\infty\text{LP}(G)$  for  $\text{Had}_4\text{-to-2Lin}(2)$ . The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints.

|                         | $\text{rsLP}(G)$  |                   |                   | $\text{rs}_\infty\text{LP}(G)$ |                   |                   |
|-------------------------|-------------------|-------------------|-------------------|--------------------------------|-------------------|-------------------|
| Original                | $1 \cdot 10^{14}$ | $3 \cdot 10^{14}$ | $4 \cdot 10^{14}$ | $1 \cdot 10^{14}$              | $3 \cdot 10^{14}$ | $4 \cdot 10^{14}$ |
| Restricted              | $2 \cdot 10^{11}$ | $4 \cdot 10^{11}$ | $6 \cdot 10^{11}$ | $2 \cdot 10^{11}$              | $4 \cdot 10^{11}$ | $6 \cdot 10^{11}$ |
| Compressed              | $4 \cdot 10^5$    | $7 \cdot 10^5$    | $1 \cdot 10^7$    | $3 \cdot 10^5$                 | $6 \cdot 10^5$    | $9 \cdot 10^6$    |
| Restricted & compressed | $3 \cdot 10^4$    | $6 \cdot 10^4$    | $2 \cdot 10^5$    | $3 \cdot 10^4$                 | $5 \cdot 10^4$    | $2 \cdot 10^5$    |

After having computed the compressed  $\text{rsLP}(G)$  and compressed  $\text{rs}_\infty\text{LP}(G)$ , the list of beneficial gadget variables found in Section 4.1 are used to construct the restricted compressed LPs. In order to solve the compressed LP we use the exact rational number LP solver `QSopt_ex`[1]. This results in a gadget described only using rational numbers, as well as an accompanying compressed flow, also described only using rational numbers.

#### E.4 Verification of $\text{rs}(G)$ and $\text{rs}_\infty(G)$

It is significantly simpler to verify the relaxed soundness and the infinity relaxed soundness of a gadget than it is to construct the gadget. The verification can be done almost directly on the original LPs, without needing the restricted compressed LPs or the compressed LPs.

The input to the verification program is a gadget  $G : \binom{\mathcal{F}_k}{2} \rightarrow [0, 1]$  together with a flow  $w_g : \mathcal{F}_k \times \mathcal{F}_k \rightarrow \mathbb{R}$ , for each source/sink placement equivalence class representative  $g$ . The flow acts as a witness for the relaxed soundness / infinity relaxed soundness of the gadget. In order to avoid floating point errors, we require both  $G$  and the  $w_g$  to be rational.

The verification process is done in five steps.

1. For each source/sink placement representative  $g$ , verify that the flow  $w_g$  satisfies the capacity constraints of the  $\text{rs}(G)$  LP /  $\text{rs}_\infty(G)$  LP, i.e. that  $w_g(f_1, f_2) + w_g(f_2, f_1) \leq G(f_1, f_2)$  for all  $f_1, f_2 \in \mathcal{F}_k$ .
2. Verify that the gadget  $G$  is symmetric under action by  $M \in \mathcal{M}_{k \rightarrow k}$ , meaning that for all functions  $f_1, f_2 \in \mathcal{F}_k$  and affine maps  $M \in \mathcal{M}_{k \rightarrow k}$ , it holds that  $G(f_1, f_2) = G(M(f_1), M(f_2))$ .
3. For each source/sink placement representative  $g$  and each function  $f \in \mathcal{F}_k$ , compute  $\text{in}(f, g)$  and  $\text{out}(f, g)$ . Now extend  $\text{in}$  and  $\text{out}$  to be defined for all  $f$  and  $g$  in  $\mathcal{F}_k$ . For any source/sink placements  $\tilde{g} \in \mathcal{F}_k$  that is not a representative, pick a map  $M \in \mathcal{M}_{k \rightarrow k}$  and representative  $g$  such that  $g = M^\#(\tilde{g})$ , and define  $\text{in}(f, \tilde{g})$  as  $\text{in}(M^{-1}(f), g)$  and  $\text{out}(f, \tilde{g})$  as  $\text{out}(M^{-1}(f), g)$ .
4. Verify the conservation of flow constraint in the  $\text{rsLP}(G)$  /  $\text{rs}_\infty\text{LP}(G)$  by iterating over all  $(f, g) \in \mathcal{F}_k \times \mathcal{F}_k$  that are not sinks or sources. For the  $\text{rsLP}(G)$  this just involves checking that  $\text{in}(f, g) = \text{out}(f, g)$ . For the  $\text{rs}_\infty\text{LP}(G)$  this involves checking that  $\sum_{g'} \text{in}(f, g') = \sum_{g'} \text{out}(f, g')$ , where the sum is over all  $g'$  such that  $g'|_{\text{affine}(f)} = g|_{\text{affine}(f)}$ .
5. Compute and output the completeness and  $\text{rs}$  /  $\text{rs}_\infty$  of the gadget using the extended inflow and outflow as a witness.

Note that the first step verifies the capacity constraints only for representatives of equivalent source/sink placements. The second step checks that the gadget  $G$  is symmetric, which combined with the first step implies that any extension of the flow to an arbitrary source/sink placement will fulfil the capacity constraints. The fourth step checks that the conservation of flow constraint is fulfilled, which in the case of the  $\text{rs}_\infty\text{LP}(G)$  involves computing the affine support of all possible source/sink placements.

The LP's we use and the gadgets we present in this paper can be found at <https://github.com/bjorn-martinsson/NP-hardness-of-Max-2Lin-2>, as well as a stand alone implementation of a verification script written in Python. As described in the verification process above, the verification requires a flow  $w_g$  as input. So on the Github, there is also a script used to generate this witness flow. This is done by solving the restricted compressed  $\text{rsLP}(G)$  /  $\text{rs}_\infty\text{LP}(G)$  using an integral Max-Flow solver, and then uncompressing the result.

**F** Edges used/unused in constructed gadgets

During the numerical analysis, we solve LPs to construct the gadgets. A gadget can be interpreted as a probability distribution over (undirected) edges. Tables 7–9 list all edges that have been given non-zero weight in at least one solution to an LP, for  $k = 2, 3, 4$ . Recall that every gadget that we construct is symmetrical under the mappings of  $\mathcal{M}_{k \rightarrow k}$ , so edges from the same edge orbit share the same capacity. More specifically, the tables contain a list of all edge orbits that are used in at least one constructed gadget.

## 11:38 On the NP-Hardness Approximation Curve for Max-2Lin(2)

■ **Table 7** The relevant edge orbits for Had<sub>2</sub>-to-2Lin(2) gadgets. The edges of a Had<sub>2</sub>-to-2Lin(2) gadget has a total of 4 edge orbits, but only two are ever used in our constructed gadgets. The rest of the edges were always given capacity 0 by the (rational) LP-solver.

| $f_1$ | $f_2$ | Ham.dist. | size |
|-------|-------|-----------|------|
| 0000  | 1000  | 1         | 32   |
| 0000  | 1100  | 2         | 24   |

■ **Table 8** The relevant edge orbits for Had<sub>3</sub>-to-2Lin(2) gadget. The edges of a Had<sub>3</sub>-to-2Lin(2) gadget has a total of 26 edge orbits, but only four are ever used in our constructed gadgets. The rest of the edges were always given capacity 0 by the (rational) LP-solver.

| $f_1$    | $f_2$    | Ham.dist. | Size |
|----------|----------|-----------|------|
| 00000000 | 10000000 | 1         | 128  |
| 10000000 | 11000000 | 1         | 896  |
| 00000000 | 11000000 | 2         | 448  |
| 00000000 | 11110000 | 4         | 112  |

■ **Table 9** The relevant edge orbits for Had<sub>4</sub>-to-2Lin(2) gadget. The edges of a Had<sub>4</sub>-to-2Lin(2) gadget has a total of 1061 edge orbits, but only 21 are ever used in our constructed gadgets. Note that as discussed in Appendix E.2.1, this list of edges was identified using the Gurobi LP-solver, and not using a rational LP solver. See Appendix E.2.1 for more information.

| $f_1$            | $f_2$            | Ham.dist. | Size   |
|------------------|------------------|-----------|--------|
| 0000000000000000 | 1000000000000000 | 1         | 512    |
| 1000000000000000 | 1100000000000000 | 1         | 7680   |
| 1100000000000000 | 1110000000000000 | 1         | 53760  |
| 1110000000000000 | 1111000000000000 | 1         | 17920  |
| 1110000000000000 | 1110100000000000 | 1         | 215040 |
| 1110100000000000 | 1110100010000000 | 1         | 215040 |
| 0000000000000000 | 1100000000000000 | 2         | 3840   |
| 1100000000000000 | 1111000000000000 | 2         | 26880  |
| 1100000000000000 | 1110100000000000 | 2         | 322560 |
| 1110000000000000 | 1111100000000000 | 2         | 107520 |
| 1110000000000000 | 1110110000000000 | 2         | 161280 |
| 1111000000000000 | 1110100000000000 | 2         | 107520 |
| 1110100000000000 | 1110100011000000 | 2         | 322560 |
| 0000000000000000 | 1110000000000000 | 3         | 17920  |
| 1100000000000000 | 1111100000000000 | 3         | 322560 |
| 1100000000000000 | 1110101000000000 | 3         | 215040 |
| 1110000000000000 | 1110100010001000 | 3         | 860160 |
| 0000000000000000 | 1111000000000000 | 4         | 4480   |
| 0000000000000000 | 1110100000000000 | 4         | 53760  |
| 0000000000000000 | 1111100000000000 | 5         | 53760  |
| 0000000000000000 | 1111111000000000 | 8         | 480    |

# Universal Optimization for Non-Clairvoyant Subadditive Joint Replenishment

**Tomer Ezra** ✉

Harvard University, Cambridge, MA, USA

**Stefano Leonardi** ✉

Sapienza University of Rome, Italy

**Michał Pawłowski** ✉

MIMUW, University of Warsaw, Poland

IDEAS NCBR, Warsaw, Poland

**Matteo Russo** ✉

Sapienza University of Rome, Italy

**Seeun William Umboh** ✉ 

School of Computing and Information Systems, The University of Melbourne, Australia

ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications

(OPTIMA), Melbourne, Australia

---

## Abstract

---

The online joint replenishment problem (JRP) is a fundamental problem in the area of online problems with delay. Over the last decade, several works have studied generalizations of JRP with different cost functions for servicing requests. Most prior works on JRP and its generalizations have focused on the clairvoyant setting. Recently, Touitou [44] developed a non-clairvoyant framework that provided an  $O(\sqrt{n \log n})$  upper bound for a wide class of generalized JRP, where  $n$  is the number of request types.

We advance the study of non-clairvoyant algorithms by providing a simpler, modular framework that matches the competitive ratio established by Touitou for the same class of generalized JRP. Our key insight is to leverage universal algorithms for Set Cover to approximate arbitrary monotone subadditive functions using a simple class of functions termed *disjoint*. This allows us to reduce the problem to several independent instances of the TCP Acknowledgement problem, for which a simple 2-competitive non-clairvoyant algorithm is known. The modularity of our framework is a major advantage as it allows us to tailor the reduction to specific problems and obtain better competitive ratios. In particular, we obtain tight  $O(\sqrt{n})$ -competitive algorithms for two significant problems: Multi-Level Aggregation and Weighted Symmetric Subadditive Joint Replenishment. We also show that, in contrast, Touitou's algorithm is  $\Omega(\sqrt{n \log n})$ -competitive for both of these problems.

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

**Keywords and phrases** Set Cover, Joint Replenishment, TCP-Acknowledgment, Subadditive Function Approximation, Multi-Level Aggregation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.12

**Category** APPROX

**Funding** *Tomer Ezra*: Supported by the Harvard University Center of Mathematical Sciences and Applications.

*Stefano Leonardi*: Supported by the ERC Advanced Grant 788893 AMDROMA and MIUR PRIN project ALGADIMAR.

*Matteo Russo*: Supported by the ERC Advanced Grant 788893 AMDROMA and MIUR PRIN project ALGADIMAR.

*Seeun William Umboh*: Part of this work was done when the author was visiting the Sapienza University of Rome, and at the School of Computer Science, University of Sydney.

**Acknowledgements** We thank the anonymous reviewers for their valuable feedback.



© Tomer Ezra, Stefano Leonardi, Michał Pawłowski, Matteo Russo, and Seeun William Umboh; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 12; pp. 12:1–12:24



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Online problems with delay have received much attention in the last few years. An important family of online problems with delay consists of the Joint Replenishment Problem (JRP) and its variants. A typical instance consists of a sequence of requests that arrive over time. Each request can be one of  $n$  request types, and the cost of serving a set of requests is a subadditive<sup>1</sup> function of their types. We assume that the algorithm has oracle access to the service cost function. Requests do not need to be served on arrival but each request accumulates a delay cost while unserved. In particular, each request  $q$  has an associated delay cost function  $d_q$  and its delay cost is  $d_q(t)$  if it is served at time  $t$ . The goal of the problem is to serve all requests minimizing the total service cost and delay cost. An important special case is the deadline case; this is when requests do not incur delay cost but instead must be served by some given time. We call this family of problems *Subadditive JRP*.

These problems can be studied under the clairvoyant and non-clairvoyant settings. In the *clairvoyant* setting, when a request  $q$  arrives, the algorithm is given the entire delay cost function  $d_q$  (or its deadline in the case of deadlines). In contrast, in the *non-clairvoyant* setting, the algorithm only knows of the delay cost accumulated so far. In the case of deadlines, the algorithm only knows whether the request's deadline is now (and must be served immediately) or later.

Most previous works on Subadditive JRP have focused on the clairvoyant setting. Key problems within the family of Subadditive JRP include (in increasing order of generality): TCP Acknowledgement [36, 27, 20], Joint Replenishment Problem [21, 18, 14, 23], and Multi-Level Aggregation (MLA) [19, 7, 12, 11, 42]. For general subadditive service cost functions, deterministic  $O(\log N)$  (where  $N$  is the number of requests) and  $O(\log n)$  upper bounds are known ([22] and [8], respectively).

There is much less work in the non-clairvoyant setting. For a small number of problems, such as TCP Acknowledgement and Set Cover with Delay [3], clairvoyance is not required in the sense that the same competitive ratio can be attained in both the clairvoyant and non-clairvoyant settings. However, Azar et al. [5]'s lower bound for Online Service with Delay (a different family of online problems with delay) can be translated into an  $\Omega(\sqrt{n})$  lower bound against deterministic algorithms for JRP, and thus, MLA and Subadditive JRP. In contrast, clairvoyant Subadditive JRP has a  $O(\log n)$  competitive ratio [8]. Recently, Le et al. [39] showed that randomization does not help in breaking the  $\Omega(\sqrt{n})$  barrier and also developed algorithms for JRP and MLA with matching and nearly-matching upper bounds. Shortly after, Touitou [44] presented a general non-clairvoyant framework for Subadditive JRP with a deterministic  $O(\sqrt{n \log n})$  competitive ratio.

### 1.1 Our Results

Our main contribution is a simple, modular framework for non-clairvoyant Subadditive JRP that matches the current-best competitive ratio of  $O(\sqrt{n \log n})$ , and yields tight  $O(\sqrt{n})$  competitive ratios for the key problems of Multi-Level Aggregation and Weighted Symmetric Subadditive Joint Replenishment. We also show that the framework of Touitou [44] is  $\Omega(\sqrt{n \log n})$  for these problems. We now formally define these problems and state our results.

---

<sup>1</sup> A set function over a ground set  $U$  is *subadditive* if  $f(A) + f(B) \geq f(A \cup B)$  for every  $A, B \subseteq U$ .



### 1.1.1 General Framework for Subadditive JRP

**Subadditive JRP.** We have a set  $U$  of  $n$  request types and a monotone non-decreasing, subadditive *service function*  $f : 2^U \mapsto \mathbb{R}_{\geq 0}$  that satisfies  $f(\emptyset) = 0$ . Requests  $q$  arrive over time. Each request  $q$  has a type  $h_q \in U$ , an arrival time  $a_q$ , and a non-decreasing, continuous delay function  $d_q$ . At any point in time, the algorithm can serve a subset  $Q$  of the requests that have arrived and incur a service cost of  $f(S_Q)$  where  $S_Q = \{h_q : q \in Q\}$  is the set of types of  $Q$ . Let  $C_q$  be the time when request  $q$  was served. The *delay cost* of request  $q$  is  $d_q(C_q)$ .<sup>2</sup> The goal is to serve all requests while minimising the sum of the total service and delay costs.

**Approximating set functions.** The core idea underlying our framework is the following simple but powerful observation. Given two set functions  $f, g$  over the same ground set  $U$  of  $n$  elements, we say that  $g$  is an  $\alpha$ -*approximation* of  $f$  if  $f(S) \leq g(S) \leq \alpha f(S)$  for every  $S \subseteq U$ . Our observation is that for a given subadditive service function  $f$ , if we can  $\alpha$ -approximate  $f$  by a simpler service function  $g$ , then we can reduce any instance of Subadditive JRP with service function  $f$  to one with  $g$  instead. In fact, this leads us to the following simplification of the problem.

**Disjoint TCP Acknowledgement.** In Disjoint TCP Acknowledgement, we have a set  $U$  of  $n$  request types. We also have a partition of  $U$  into subsets  $S_1, \dots, S_k$  with costs  $c_1, \dots, c_k$ . For a subset  $S \subseteq U$ , we have  $f(S) = \sum_{i=1}^k c_i \cdot \mathbb{1}\{S_i \cap S \neq \emptyset\}$ . In other words, we pay  $c_i$  for every part  $S_i$  that intersects with  $S$ . Such a function is called a *disjoint service function*. Observe that when  $k = 1$ , this is equivalent to the TCP Acknowledgement problem; when  $k > 1$ , this corresponds to several independent instances of TCP Acknowledgement. The 2-competitive algorithm for TCP Acknowledgement of [27] can be easily extended to a 2-competitive algorithm for Disjoint TCP Acknowledgement (see Section 2.1).

We now state our main technical lemma.

► **Lemma 1.1** (Reduction Lemma). *If there exists a **disjoint** service function  $g$  that  $\alpha$ -approximates  $f$ , then there exists a non-clairvoyant algorithm that is  $2\alpha$ -competitive non-clairvoyant algorithm for every Subadditive JRP instance with service cost function  $f$ .*

A major advantage of our Reduction Lemma is that it reduces the task of designing and analyzing an online algorithm for a Subadditive JRP problem to the much cleaner task of showing that the corresponding service function  $f$  can be approximated by a disjoint service function well. In particular, this boils down to finding a partition of the set of request types  $U$  into subsets  $S_1, \dots, S_k$ , for some  $k$ , such that the following quantity is small

$$\max_{S \subseteq U} \frac{\sum_{i=1}^k f(S_i) \cdot \mathbb{1}\{S_i \cap S \neq \emptyset\}}{f(S)}.$$

For general Subadditive JRP, our key insight is that the problem of approximating an arbitrary service function  $f$  by a disjoint service function can be reformulated as the Universal Set Cover problem.

<sup>2</sup> We assume W.L.O.G. that  $d_q(a_q) = 0$ , i.e., serving a request immediately on arrival incurs no delay cost.

**Universal Set Cover (USC).** An instance of the Universal Set Cover (USC) problem consists of a universe  $U$  of  $n$  elements, a collection  $\mathcal{C}$  of subsets of  $U$ , and costs  $c(S)$  for each set  $S \in \mathcal{C}$ . A solution is an assignment  $a$  of each element  $e$  to a set  $a(e) \in \mathcal{C}$ . For any subset  $X \subseteq U$ , define  $a(X) = \{a(e) : e \in X\}$ . The *stretch* of the assignment  $a$  is  $\max_{X \subseteq U} c(a(X))/\text{OPT}(X)$  where  $\text{OPT}(X)$  is the cost of the optimal set cover of  $X$ .

Jia et al. [35] introduced the Universal Set Cover problem and showed that a  $O(\sqrt{n \log n})$ -stretch assignment can always be found efficiently. We show that this implies that any subadditive service function  $f$  can be approximated by a disjoint service function to within a factor of  $O(\sqrt{n \log n})$  (Lemma 2.2). Together with our Reduction Lemma, we get a deterministic  $O(\sqrt{n \log n})$ -competitive algorithm for Non-Clairvoyant Subadditive JRP, matching the current state-of-the-art [44].

### 1.1.2 MLA and Weighted Symmetric Subadditive JRP

One main technical contribution of the paper is to exploit the inherent structure of the MLA and Weighted Symmetric Subadditive JRP functions to show that they can be  $O(\sqrt{n})$ -approximated by disjoint service functions. We then employ the Reduction Lemma to prove tight  $O(\sqrt{n})$ -competitive ratios for the two corresponding problems.

**Multi-Level Aggregation.** In the Multi-Level Aggregation (MLA) problem, the service function  $f$  is defined by a rooted *aggregation tree*  $T$ , where each node corresponds to a different request type. Let  $r$  be the root of  $T$  and let  $c(v)$  be the cost of node  $v$  for each  $v \in T$ . For a subset  $V$  of nodes,  $f(V)$  is defined to be the total cost of the nodes in the minimal subtree connecting  $V$  to  $r$ .

► **Theorem 1.2.** *There exists an efficient deterministic  $O(\sqrt{n})$ -competitive algorithm for the Non-Clairvoyant Multi-Level Aggregation problem.*

To show the above result, given Lemma 1.1, our goal is to find a good partition  $P$  of the tree  $T$ 's nodes into subtrees and subforests (that we refer to as clusters). More precisely, let us use  $P$  to define a disjoint service function  $g$  where for each subset  $V$  of nodes of  $T$ ,  $g(V) = \sum_{C \in P: C \cap V \neq \emptyset} f(C)$ .

The crucial idea is to notice that since we aim for the gap of order at most  $\sqrt{n}$  between  $g$  and  $f$ , we can see it as  $g$  being assigned a budget of roughly  $\sqrt{n}f(V)$  to serve  $V$  for each subset  $V$  of  $T$ 's nodes. Since the cost that  $f$  incurred on a set  $V$  equals the cost of the minimal subtree connecting all the nodes in  $V$  to the root  $r$  of  $T$ , the value of  $g(V)$  cannot exceed  $\beta\sqrt{n}$  times this cost for some fixed  $\beta \in \mathbb{N}$ . To achieve this, we generate a partition consisting of two types of clusters. First are the subtrees rooted at “expensive” nodes. The intuition is that their cost alone multiplied by  $\alpha\sqrt{n}$  for some  $\alpha \in \mathbb{N}$  is enough to “cover” the cost of both their subtree and the path to  $r$ . The second type is the clusters that contain more than  $\sqrt{n}$  nodes, since there cannot be many of them.

**Weighted Symmetric Subadditive JRP.** In Weighted Symmetric Subadditive JRP, the service function  $f$  is a function of the total weight  $w(S) = \sum_{i \in S} w_i$  of the set of types being served. In particular,  $f$  is a monotone non-decreasing subadditive function with  $f(S) = f(w(S))$  and  $f(0) = 0$ , that satisfies that for every weights  $x, y$ ,  $f(x+y) \leq f(x) + f(y)$ . We refer to these functions as *weighted symmetric subadditive*.

► **Theorem 1.3.** *There exists an efficient deterministic  $O(\sqrt{n})$ -competitive algorithm for the Non-Clairvoyant Weighted Symmetric Subadditive Joint Replenishment problem.*

As in the MLA case, given Lemma 1.1, our goal is to devise a partitioning algorithm inducing a disjoint service function that  $O(\sqrt{n})$ -approximates the corresponding weighted symmetric subadditive service cost function. We first consider the special case where each weight equals 1. In this scenario, the service function  $f$  is symmetric and becomes a function of the cardinality of the set of types being served. Consequently, the partition of elements should ideally reflect this symmetry by ensuring equal-sized parts.

Determining the optimal size for each part involves striking a delicate balance. Larger sizes enable us to leverage the subadditivity of  $f$  but excessively large sizes incur higher costs for smaller sets. We demonstrate that selecting sets of size  $O(\sqrt{n})$  is the optimal tradeoff in worst-case scenarios. Notably, this partition remains effective across all symmetric subadditive functions simultaneously.

Extending this approach to the general case of weighted symmetric subadditive functions involves categorizing elements into weight classes based on powers of 2, ensuring approximate size equivalence, and then partitioning into sets of size  $\sqrt{n}$ . However, this approach risks generating an excessive number of sets. To address this issue, we devise a partitioning strategy that accommodates light-weight elements first. Then, for heavier-weight elements, we further partition by a factor of 2, provided it is feasible, to achieve a refined division.

### 1.1.3 Running time of Algorithms and Reductions

Regarding the running time of our algorithms, we stress that, in the case of Multi-Level Aggregation and Weighted Symmetric Subadditive JRP, the reductions are executed in polynomial time. However, the reduction for general subadditive functions is executed in exponential time, as we need to create a set for each subset of types.

### 1.1.4 Lower bounds on approximating subadditive service functions

Since Non-Clairvoyant MLA and Weighted Symmetric Subadditive JRP have a  $\Omega(\sqrt{n})$  lower bound [5, 39], the Reduction Lemma implies that MLA and Weighted Symmetric Subadditive JRP service functions do not admit  $o(\sqrt{n})$ -approximation by disjoint service functions. Nevertheless, we also give direct proofs in Sections 3 and 4, respectively. The latter provides a simpler alternative proof for the  $\Omega(\sqrt{n})$  lower bound for unweighted Universal Set Cover shown in [35]. We also show, in Proposition 5.1, that Jia et al.’s analysis of their Universal Set Cover algorithm [35] is tight. Thus, we need a different approach to  $o(\sqrt{n \log n})$ -approximate arbitrary subadditive service functions by disjoint service functions. Finally, in Proposition 5.2, we exhibit an MLA and Weighted Symmetric Subadditive JRP instance where Touitou’s algorithm [44] can only achieve an  $\Omega(\sqrt{n \log n})$ -approximation to the respective service cost functions.

## 1.2 Future Directions

Our work leaves several tantalizing open questions. The main open problem is whether subadditive service functions admit better than  $O(\sqrt{n \log n})$ -approximation by disjoint service functions. This would immediately improve the competitive ratio for general non-clairvoyant Subadditive JRP. It would also be interesting to find better approximations of other interesting subclasses such as XOS and submodular functions.

### 1.3 Further Related Work

**Network Design with Delay.** Network Design with Delay is very closely related to Subadditive JRP. In Network Design with Delay, we are given a universe of  $n$  request types and  $m$  items with costs. Each request type  $h$  has a corresponding upwards-closed collection  $\mathcal{C}_h$  of subsets of items that satisfy it. At any point in time, the algorithm can transmit a set of items. A request of type  $h$  is served by a transmission that contains some subset in  $\mathcal{C}_h$ . Some specific problems are Set Cover with Delay [22, 3, 43], Facility Location [7, 8, 13] and other network design problems [7, 8]. Network Design with Delay is equivalent to Subadditive JRP as the optimal cost of satisfying a subset of request types is subadditive, and Subadditive JRP can be formulated as Set Cover with Delay with exponentially many sets.

**Online problems with delay.** There has been a lot of work on other online problems with delay as well. In Online Service with Delay, we are given one or multiple servers on a metric space. Requests arrive on points of the metric space and are served when a server is moved to their location. In Online Matching with Delay, we are given an underlying metric space. Requests arrive on points of the metric space and are served when they are matched. Most of the work on Online Service with Delay [5, 33, 34, 17, 38, 45] and Online Matching with Delay [28, 2, 1, 16, 4, 15, 29, 6, 41, 24] has been in the clairvoyant setting. Nevertheless, non-clairvoyant algorithms have been designed for Online Service with Delay [38] and Online Matching with Delay [24].

**Approximating subadditive functions.** The approximation of subadditive functions has been a focal point of research, at least since the introduction of the complement-free hierarchy of functions introduced in [40]. This consists of the class of submodular function, which is strictly contained into the XOS class, which in turn is strictly contained in the general subadditive class.<sup>3</sup> As for approximation, it is known that XOS approximates subadditive within a factor of  $O(\log(n))$ , which is tight [25, 10]. The approximability gap between Submodular and XOS is  $\Theta(\sqrt{n})$  [9, 31]. In a similar vein, [26] prove that Gross-Substitute functions (first introduced in [37]) cannot approximate submodular set functions within a factor better than  $\Omega\left(\frac{\log(n)}{\log \log(n)}\right)$ . In the context of *symmetric* function approximation, [30] show that symmetric subadditive, symmetric XOS and symmetric submodular<sup>4</sup> functions are all 2-close to each other, which is tight.

## 2 Subadditive Joint Replenishment

In this section, we prove our Reduction Lemma (Lemma 1.1) and apply it to Subadditive JRP.

### 2.1 Reduction Lemma

We begin by showing that there is a simple deterministic 2-competitive algorithm for Disjoint TCP Acknowledgement via a straightforward extension of the classic algorithm for TCP Acknowledgement of [27].

<sup>3</sup> Several other classes within the submodular class have been considered (e.g. additive, unit-demand, Gross-Substitutes).

<sup>4</sup> We use the term symmetric submodular to indicate functions that are (monotone) concave in the size of the set.

In the following, we use  $\lambda$  to denote a service and  $Q_\lambda$  to be the set of request types transmitted by  $\lambda$ . We also use OPT to mean both the optimal solution and the cost of the optimal solution.

► **Lemma 2.1.** *There is a deterministic 2-competitive algorithm for Disjoint TCP Acknowledgement.*

**Proof.** Suppose there is a partition of  $H$  into subsets  $S_1, \dots, S_k$  with costs  $c_1, \dots, c_k$  and  $f(S) = \sum_{i=1}^k c_i \cdot \mathbb{1}\{S_i \cap S \neq \emptyset\}$ . Our algorithm works as follows: for each set  $S_i$ , transmit  $S_i$  whenever the pending requests in  $S_i$  have accumulated a total delay equal to  $c_i$ .

It is clear that the total service cost of the algorithm is at most its total delay cost. We now show that the latter is at most the cost of the optimal solution. To this end, let us consider the cost of the optimal solution. Suppose that the optimal solution makes a set of services  $\Lambda^*$ . Let  $\Lambda_i^*$  denote the subset of services that transmit a request type in  $S_i$ . The total service cost of the optimal solution is then

$$\sum_{\lambda \in \Lambda^*} f(Q_\lambda) = \sum_{\lambda \in \Lambda^*} \sum_{i=1}^k c_i \cdot \mathbb{1}\{S_i \cap Q_\lambda \neq \emptyset\} = \sum_{i=1}^k c_i \cdot |\Lambda_i^*|.$$

Define  $d_q^{\text{OPT}}$  and  $d_q^{\text{ALG}}$  to be the delay cost of  $q$  in the optimal solution and algorithm's solution, respectively. Let  $\text{OPT}_i = c_i \cdot |\Lambda_i^*| + \sum_{q: h_q \in S_i} d_q^{\text{OPT}}$ . This is the total cost that OPT incurs on requests on  $S_i$ . Observe that  $\text{OPT} = \sum_{i=1}^k \text{OPT}_i$ .

We now show that  $\sum_{q: h_q \in S_i} d_q^{\text{ALG}} \leq \text{OPT}_i$  for each set  $S_i$ . Suppose that the algorithm transmits  $S_i$  at times  $t_1, \dots, t_\ell$ . Since every request must be served eventually, no request with type in  $S_i$  arrives after  $t_\ell$ . Consider the intervals  $[0, t_1], (t_1, t_2], \dots, (t_{\ell-1}, t_\ell]$ . By construction, the delay cost of the algorithm is exactly  $\ell c_i$ . For each interval  $I$ , let  $Q(I)$  denote the requests with types in  $S_i$  that arrived during the interval. During  $I$ , the optimal solution either transmits a type in  $S_i$  or incurs a delay cost of  $c_i$  on the requests in  $Q(I)$ . Since the intervals are disjoint,  $\text{OPT}_i \geq \ell c_i$ , as desired.

The lemma now follows from the fact that the total service cost of the algorithm is exactly its delay cost, which in turn is at most OPT. ◀

We are now ready to prove the Reduction Lemma which we restate here.

► **Lemma 1.1 (Reduction Lemma).** *If there exists a **disjoint** service function  $g$  that  $\alpha$ -approximates  $f$ , then there exists a non-clairvoyant algorithm that is  $2\alpha$ -competitive non-clairvoyant algorithm for every Subadditive JRP instance with service cost function  $f$ .*

**Proof.** Lemma 2.1 implies that it suffices to reduce the Subadditive JRP instance to an instance of Disjoint TCP Acknowledgement losing at most a factor of  $\alpha$ . Let  $Q$  be the set of requests of the Subadditive JRP instance and let  $\text{OPT}^f$  denote the cost of the optimal solution. Our reduction creates an instance of Disjoint TCP Acknowledgement with the same set of requests but with service cost function  $g$ . Let  $\text{OPT}^g$  denote the cost of the optimal solution to the instance of Disjoint TCP Acknowledgement. We now show that  $\text{OPT}^f \leq \text{OPT}^g \leq \alpha \text{OPT}^f$ . Let  $\Lambda$  be a feasible solution to  $Q$ ,  $c^f(\Lambda)$  be its cost in the Subadditive JRP instance and  $c^g(\Lambda)$  be its cost in the Disjoint TCP Acknowledgement instance. The delay cost of  $\Lambda$  is the same in both instances. The service cost of  $\Lambda$  in the Subadditive JRP instance has cost  $\sum_{\lambda \in \Lambda} f(Q_\lambda)$  and in the Disjoint TCP Acknowledgement instance, it has cost  $\sum_{\lambda \in \Lambda} g(Q_\lambda)$ . Since  $g$   $\alpha$ -approximates  $f$ , we get that  $c^f(\Lambda) \leq c^g(\Lambda) \leq \alpha c^f(\Lambda)$ . This implies that  $\text{OPT}^f \leq \text{OPT}^g \leq \alpha \text{OPT}^f$ , as desired. ◀

## 2.2 Applying the Reduction Lemma to Subadditive JRP

We use the Reduction Lemma proved earlier to give a simple deterministic  $O(\sqrt{n \log n})$ -competitive algorithm for Non-Clairvoyant Subadditive JRP. The main insight is to reduce the problem of showing that an arbitrary service function  $f$  can be approximated by a disjoint service function to the Universal Set Cover problem.

► **Lemma 2.2.** *Suppose every instance of USC admits a  $\alpha$ -stretch assignment. Then every subadditive service function  $f$  can be  $\alpha$ -approximated by some disjoint service function  $g$ .*

**Proof.** We will construct an instance of USC and use the  $\alpha$ -stretch assignment to construct  $g$ . Consider the instance of USC with universe  $U = H$ ,  $\mathcal{C} = 2^H$ , and  $c(S) = f(S)$  for every  $S \in \mathcal{C}$ . Note that  $\text{OPT}(S) = f(S)$  since  $f$  is monotone non-decreasing and subadditive.

Let  $a$  be an  $\alpha$ -stretch assignment for this USC instance. Suppose  $a(U) = \{S_1, \dots, S_k\}$ . Since  $a$  maps each element to a set containing it, we have that  $a^{-1}(S_i) \subseteq S_i$ . Moreover,  $f$  is monotone non-decreasing, so we can assume W.L.O.G. that  $a^{-1}(S_i) = S_i$ ;<sup>5</sup> thus  $S_1, \dots, S_k$  are disjoint and partition  $H$ . Define the disjoint service function  $g$  with the partition  $\{S_1, \dots, S_k\}$  and costs  $c_1, \dots, c_k$  where  $c_i = f(S_i)$ . Observe that  $g(S) = c(a(S)) \geq \text{OPT}(S) = f(S)$ . Since  $a$  has  $\alpha$ -stretch, we get that for every  $S$ ,  $f(S) \leq g(S) \leq \alpha f(S)$ . ◀

Jia et al. [35] showed that every instance of USC has a  $O(\sqrt{n \log n})$ -stretch assignment. Together with the above lemma, we get the following theorem.

► **Theorem 2.3.** *For every subadditive service function  $f$ , there is a disjoint service function  $g$  that  $O(\sqrt{n \log n})$ -approximates  $f$ .*

Combining this with the Reduction Lemma yields the desired theorem.

► **Theorem 2.4.** *There is a deterministic  $O(\sqrt{n \log n})$ -competitive algorithm for Non-Clairvoyant Subadditive JRP.*

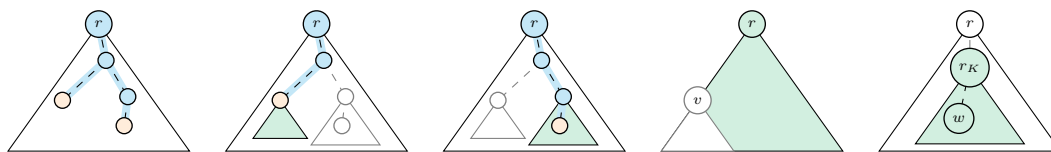
## 3 Multi-Level Aggregation

In this section, we consider the Multi-Level Aggregation (MLA) problem. Let  $T = (U, E)$  be a rooted tree defined over the universe  $U$  of  $n$  request types and let  $c : U \mapsto \mathbb{R}_{\geq 0}$  be a cost function assigning weights to the nodes. We recall that  $c$  determines the service function  $f : 2^U \mapsto \mathbb{R}_{\geq 0}$  for this problem as  $f$  assigns to each subset of nodes  $V \subseteq U$  the cost of the minimal subtree that connects all the nodes in  $V$  to the root  $r$ . Here, we prove that for every MLA service function  $f$ , there exists a disjoint service function  $g : 2^U \mapsto \mathbb{R}_{\geq 0}$  that  $O(\sqrt{n})$ -approximates  $f$ . In other words, we show that for every MLA instance  $(T, c)$ , there exists a partition  $P_1, \dots, P_k$  of nodes of  $T$  for some  $k$  (which defines  $g(X) = \sum_{i \in [k]} f(P_i) \cdot \mathbb{1}\{P_i \cap X \neq \emptyset\}$  for all  $X \subseteq U$ ), such that for all  $V \subseteq U$ , it holds that  $g(V)/f(V) \leq O(\sqrt{n})$ . Moreover, one can find such a partition in polynomial time.

### 3.1 Notation and Algorithm Overview

Throughout this section, we assume tree  $T$  is the current MLA instance that we work with and thus is known from the context. In what follows, we refer to the *maximal subtree* of  $T$  rooted at node  $v$  and to the *path* connecting  $v$  to the root  $r$  by simply writing  $T(v)$  and  $R(v)$ , respectively. Moreover, to denote these objects with node  $v$  excluded, we use  $T_o(v)$  and  $R_o(v)$ . Finally, we let  $C(v)$  be the set of  $v$ 's *children* in  $T$ .

<sup>5</sup> Otherwise, we can assign the elements in the preimage of  $S_i$  under  $a$ , i.e.,  $a^{-1}(S_i)$ , to the preimage itself.



■ **Figure 1** In the first three figures we show the costs of serving the orange nodes. The first figure corresponds to the cost of  $f$  on these nodes. The following two figures show the cost of  $g$  on these nodes, assuming that they belong to different clusters  $A$  and  $B$ . Fourth figure shows the set of active nodes in the tree (colored in green) after  $T(v)$  gets clustered. Fifth figure presents the setting in Proposition 3.7.

First, we present the idea behind our approach. Recall that our goal is to find a partition  $P_1, \dots, P_k$  of nodes of  $T$  such that the gap between the values  $f(V)$  and  $g(V)$  is at most of order  $\sqrt{n}$  for all the sets  $V \subseteq U$ . Here,  $f(V)$  is the cost of minimal subtree  $T_V$  of  $T$  that connects all the nodes in  $V$  to the root, as stated before. On the other hand,  $g(V)$  needs to cover the costs of all the parts in  $P$  that intersect with  $V$ . For instance, if  $V$  intersects exactly two parts  $A$  and  $B$  in  $P$ , then  $g(V) = f(A) + f(B)$ . Although these parts themselves are disjoint by the definition of partition  $P$ , as we pay for each of them separately in  $g$  (by paying for set  $A$ , we mean generating the cost of  $f(A)$ ), we incur not only the costs of their nodes  $c(A)$  and  $c(B)$  but also the costs of the paths that connect them to the root  $r$  (see Figure 1).

Note that this process can cause us to incur two types of additional costs with respect to the optimal value  $f(V)$ . First, both parts  $A$  and  $B$  may contain not only the nodes in  $V$  but also their neighbors, for which we need to pay as well. Second, as we pay for each part separately, we may be forced to pay for some nodes on the paths to the root multiple times (see Figure 1).

Since  $f(V)$  is equal to the cost of the nodes in  $T_V$  and we aim for  $g$  to be  $\sqrt{n}$ -approximation of  $f$ , the intuition is that  $g$  can afford to pay the cost of each node in  $T_V$  roughly  $\sqrt{n}$  times (as this gives the desired ratio). This observation provides the foundations for our algorithm. Let us remark that at the beginning, all the nodes in  $T$  are unpartitioned, i.e.,  $P = \emptyset$ . Our algorithm revolves around two procedures. The first one can be seen as assigning each node  $v$  in  $T$  a budget of  $\alpha\sqrt{n} \cdot c(v)$  for some  $\alpha \in \mathbb{N}$ . A vertex  $v$  may then use such a budget to create a new part  $K$  in the partition. We allow  $v$  to generate only one form of a *cluster*, i.e., a part to be included in  $P$ , that consists of all the unpartitioned nodes in its subtree  $T(v)$ . Furthermore, for such a part  $K$  to be added to  $P$ , it needs to hold that the costs of (the unpartitioned nodes in)  $T(v)$  and  $R(v)$  both fit into  $v$ 's budget. If we manage to add  $K$  to  $P$ , we call both node  $v$  and cluster  $K$  *heavy*.

Whenever the first procedure cannot be applied, i.e., there are no vertices that can generate heavy clusters from the unpartitioned nodes, we run the second procedure. The idea then is to find a subtree (or a family of subtrees) of size roughly  $\sqrt{n}$  (details to be presented later) and group them together into a new part in the partition. We call this part a *light cluster*. In case there are nodes that become heavy after this action (as their descendants got clustered), we go back to the first procedure, which starts a new iteration of the main algorithm.

Notice that the idea behind the second procedure is to upper bound the number of times we need to pay the cost of the paths connecting the clusters to the root  $r$ . Since  $T$  has  $n$  nodes and each light cluster is of size close to  $\sqrt{n}$ , we can only create roughly  $\sqrt{n}$  such clusters. Thus, even when  $V$  intersects all the light clusters, we pay for the nodes in  $T_V$  at most  $O(\sqrt{n})$  times, which we can afford. It remains to estimate the cluster costs, which follow in the next section.

### 3.2 MLA Partitioning Algorithm

**Heavy clusters.** Let us first present two definitions. Here, we assume that whenever we are given a partially created partition  $\tilde{P}$  of nodes of an MLA tree  $T$ , then the set of nodes it already partitioned, i.e.,  $V(\tilde{P}) = \bigcup \tilde{P}$ , does not disconnect  $T$ , i.e., tree  $T' = T \setminus \bigcup \tilde{P}$  is a subtree of  $T$ .

► **Definition 3.1.** Let  $T$  be a tree given in an MLA instance, denote the set of its nodes by  $U$ , and let  $\tilde{P}$  be a partially created partition of  $U$  (i.e.,  $V(\tilde{P}) = \bigcup \tilde{P}$  is a proper subset of  $U$ ). Then we call all the nodes that are not partitioned yet, i.e., belong to  $U \setminus V(\tilde{P})$ , active (see Figure 1). We use the notation of  $V|_{act}$  to restrict any subset  $V$  of nodes in  $T$  to the nodes that are active.

► **Definition 3.2.** Let  $(T, c)$  be an MLA instance, and let  $\tilde{P}$  be a partially created partition of  $T$ 's nodes. We say that an active node  $v$  is heavy if the costs of path  $R(v)$  and subtree  $T_{act}(v)$  are at most  $4\sqrt{n} \cdot c(v)$  each. If we extend  $\tilde{P}$  by adding  $T_{act}(v)$ , we call this new part a heavy cluster.

Now, we can prove a simple fact about heavy clusters.

► **Proposition 3.3.** Let  $(T, c)$  be an MLA instance. Take any partition  $P$  of nodes of  $T$  and let  $P_{h,1}, \dots, P_{h,s}$  be a sublist of all heavy clusters in  $P$ . We denote their roots by  $v_{h,1}, \dots, v_{h,s}$ , respectively, and the set containing them by  $V_h$ . Then, it holds that  $\sum_{i=1}^s f(P_{h,i}) \leq 8\sqrt{n} \cdot f(V_h)$ .

**Proof.** By Definition 3.2, we have that for each node  $v_{h,i}$  the following is satisfied:  $c(P_{h,i}) \leq 4\sqrt{n} \cdot c(v_{h,i})$  and  $c(R(v_{h,i})) \leq 4\sqrt{n} \cdot c(v_{h,i})$ . The first inequality here comes from the fact that cluster  $P_{h,i}$  was the set of all active nodes (Definition 3.1) contained in the subtree  $T(v_{h,i})$  at the moment it was created (i.e., it was  $T_{act}(v_{h,i})$ ). Hence, it holds that

$$\begin{aligned} f(P_{h,i}) &= c(P_{h,i}) + c(R(v_{h,i})) \leq c(P_{h,i}) + c(R(v_{h,i})) \\ &\leq 4\sqrt{n} \cdot c(v_{h,i}) + 4\sqrt{n} \cdot c(v_{h,i}) = 8\sqrt{n} \cdot c(v_{h,i}), \end{aligned} \quad (1)$$

where the first equality comes from the fact that  $P_{h,i}$  is a subtree, which means that the minimal tree containing all its nodes and the root  $r$  is only missing the path from  $v$  to  $r$  (with  $v$  excluded as we already counted it in the cluster). Moreover, let us notice that  $f(V_h) \geq \sum_{i=1}^s c(v_{h,i})$ , as it is the cost of the minimal tree containing all the nodes  $v_{h,i}$ . Thus, to obtain the desired inequality, we only need to sum (1) over all the heavy clusters and then apply the inequality above. ◀

**Light clusters.** Here, we present a procedure that generates a light cluster.

► **Definition 3.4.** Let  $(T, c)$  be an MLA instance, and let  $\tilde{P}$  be a partially created partition of  $T$ 's nodes. We say that a subset  $K$  of nodes of  $T_{act}$  is a light cluster if (1) its size fits into the range  $I(n) := [\sqrt{n}, 2\sqrt{n}]$ , (2) it is either a maximal subtree in  $T_{act}$  or a collection of maximal subtrees having the same parent, and (3)  $T_{act}$  does not contain any heavy nodes.<sup>6</sup> In case  $T_{act}$  is of size smaller than  $\sqrt{n}$ , and we set  $K = T_{act}$ , we drop the first condition and still call  $K$  a light cluster.

Given the definition above, we present Algorithm 1 that shows how to find such a cluster.

<sup>6</sup> This third condition is for analysis purposes only and the property giving the name to *light* clusters.



■ **Algorithm 1** MLA Light Cluster Search.

**Input:** MLA tree  $T$  with some nodes marked active ( $T_{act}$  is a subtree of  $T$  containing its root  $r$ )

**Output:** light cluster formed of the nodes in  $T_{act}$

---

```

1: if  $|T_{act}| \leq 2\sqrt{n}$ 
2:   return  $T_{act}$ 
3:  $u := r$ 
4: while there exist a node  $v \in C_{act}(u)$  such that  $|T_{act}(v)| > 2\sqrt{n}$ 
5:    $u := v$ 
6: if there exist a node  $v \in C_{act}(u)$  such that  $|T_{act}(v)| \geq \sqrt{n}$  then
7:   return  $T_{act}(v)$ 
8: else
9:   denote all the elements in  $C_{act}(u)$  by  $v_1, v_2, \dots, v_j$  for some  $j$ 
10:  set iterator  $i = 1$  and initialize a new cluster  $V$  with an empty set
11:  while  $|V| < \sqrt{n}$ 
12:    add  $T_{act}(v_i)$  to  $V$ 
13:    increment  $i$  by 1
14:  return  $V$ 

```

---

► **Proposition 3.5.** *If there are no heavy nodes in  $T_{act}$  (see condition (3) in Definition 3.4), then Algorithm 1 finds a light cluster in  $T_{act}$ .*

**Proof.** Notice that we start the search of a new cluster by checking whether the size of  $T_{act}$  (the subtree containing all the active nodes in  $T$ ) is smaller or equal to  $2\sqrt{n}$  (line 1). If so, we return the whole tree  $T_{act}$  since it fits into the description given in the last sentence of Definition 3.4. Otherwise, we set  $r$  to be the current node we are at, which we denote by  $u$  (line 3). Then, we go through the while loop from line 4 to 5, each time picking a child  $v$  of the current node  $u$  such that the subtree  $T_{act}(v)$  is of size greater than  $2\sqrt{n}$ . If such a node exists, we move to it, setting  $u = v$ , and we leave the while loop otherwise.

In the second case, we know that, as we go to line 6, two conditions are satisfied. First, the size of the subtree  $T_{act}(u)$  rooted at the current node  $u$  is at least  $2\sqrt{n}$ . Indeed, we either stayed at the root node, not satisfying the condition in the if statement in line 1, or we further went from  $r$  through a sequence of its descendants, each having a subtree of size greater than  $2\sqrt{n}$ . Second, none of  $u$ 's children has a subtree of size greater than  $2\sqrt{n}$ , as we already left the while loop.

Now, in line 6, we check whether there exists a child  $v$  of the current node, which subtree  $T_{act}(v)$  is of size at least  $\sqrt{n}$ . If so, we return  $T_{act}(v)$ , as it satisfies the conditions to be a light cluster. Otherwise (line 8), we iterate through  $u$ 's children  $v_i$  (line 11) and add the nodes contained in their subtrees  $T_{act}(v_i)$  to a set  $V$ . We stop at the moment when the size of  $V$  becomes at least  $\sqrt{n}$  and return  $V$  as a new cluster. It is easy to notice that in the while loop, we indeed need to pass the  $\sqrt{n}$  size threshold, as  $|T_{act}(u)| > 2\sqrt{n}$ . Moreover, we know that before we added the nodes of the last subtree  $T'$  to  $V$ ,  $V$  had a size smaller than  $\sqrt{n}$ . Since  $|T'| < \sqrt{n}$ , we have that the whole group is of size smaller than  $2\sqrt{n}$ . ◀

**Main algorithm.** Before we describe the partitioning algorithm, let us introduce a helper function. We define method `cluster(V)` to group all the elements of  $V$  together and include them as a new part in the partition. Let us also emphasize that after this call, all the elements in  $V$  become inactive. With the above notation, we can formalize our approach as presented in Algorithm 2.

---

**Algorithm 2** MLA Partitioning Algorithm.

---

**Input:** MLA instance  $(T, c)$ **Output:** partition  $P$  of the nodes of  $T$ 

```

1: initialize an empty partition  $P$ 
2: while  $T_{act}$  is not empty
3:   while there exist a heavy node  $v \in T_{act}$ 
4:     cluster $(T_{act}(v))$ 
5:   if  $T_{act}$  is empty
6:     break
7:   apply Algorithm 1 to find a light cluster  $V$  in  $T_{act}$ 
8:   cluster $(V)$ 
9: return  $P$ 

```

---

As mentioned in the first part of this section, the main partitioning algorithm runs heavy and light cluster searches in a loop. In the first step, it iteratively finds the heavy clusters in the tree  $T_{act}$  determined by the already created partition (lines 3 to 4). Then, if tree  $T$  is not yet partitioned (condition in line 5 does not hold), it goes to the second step that finds one light cluster and adds it to the partition (lines 7 to 8). After this point, it goes to the initial step and loops.

Let us emphasize that during the whole partitioning procedure, the set  $T_{act}$  of all active elements in  $T$  forms a proper subtree containing the root  $r$  of  $T$ . Indeed, in the beginning,  $T_{act} = T$  and all the **cluster** calls truncate one or more maximal subtrees from  $T_{act}$ . Now, given Algorithm 2, we go back to proving the properties of light clusters.

► **Proposition 3.6.** *Let  $T$  be an MLA tree rooted at some node  $r$  and let  $P$  be the partition of nodes of  $T$  created by Algorithm 2. We denote all the light clusters in  $P$  by  $P_{\ell,1}, P_{\ell,2}, \dots, P_{\ell,t}$  and require them to be listed in the creation order. Then, it holds that there are at most  $\sqrt{n} + 1$  parts  $P_{\ell,i}$ .*

**Proof.** Notice that by the definition, the only light cluster that can have a size smaller than  $\sqrt{n}$  is the one containing the root  $r$ . Thus, all the light clusters created before, i.e., at least  $t - 1$  of them, have the size at least  $\sqrt{n}$ . Since there are  $n$  nodes in tree  $T$ , we get that there are at most  $n/\sqrt{n} = \sqrt{n}$  such clusters. Thus,  $t \leq \sqrt{n} + 1$ , which concludes the proof. ◀

In the remaining part of this section, we refer to the clusters created in lines 2, 7 of Algorithm 2, i.e., the ones that consist of a single subtree, as the light clusters of type I. We call the light clusters consisting of forests (created in line 14) the light clusters of type II. We prove that the cost function  $c$  satisfies the following properties. Here, we overuse the notation of  $c$  and extend it to the subsets as well, i.e., for any  $V \subseteq U$  we set  $c(V) = \sum_{v \in V} c(v)$ .

► **Proposition 3.7.** *Let  $(T, c)$  be an MLA instance and let  $P$  be the partition obtained on it by Algorithm 2. Take any light cluster  $K$  in  $P$  and denote by  $r_K$  the root of  $K$  if it is a cluster of type I. Otherwise, if  $K$  is a cluster of type II, we use  $r_K$  to denote the parent node of the forest contained in  $K$ . Then it holds that  $c(P(r_K)) \geq c(K)$ .*

**Proof.** Without loss of generality, assume that  $K$  is of type I. Let  $w$  be the node in  $K$  that has the highest cost. By Definition 3.4, we know that  $|K| \leq 2\sqrt{n}$ . Hence, by an averaging argument, we have  $c(w) \geq c(K)/(2\sqrt{n})$ , which implies  $2\sqrt{n} \cdot c(w) \geq c(K)$ . Now, assume by contradiction that  $c(P(r(K))) < c(K)$ . Then, if we split the path from  $w$  to  $r$  into two parts by cutting it on the node  $r_K$ , we got  $c(P(w)) = c(P(w) \cap K) + c(P_o(r_K))$ . Since  $c(P(w) \cap K) \leq c(K)$  and  $c(P(r_K)) \leq c(K)$  by our assumption, we get that  $c(P(w)) \leq$

$2c(K) \leq 2 \cdot 2\sqrt{n} \cdot c(w) = 4\sqrt{n} \cdot c(w)$ . However, this means that  $w$  is a heavy node, which contradicts the initial assumption. Thus, it holds that  $c(P(r_K)) \geq c(K)$ . The proof for type II follows the same steps.  $\blacktriangleleft$

► **Corollary 3.8.** *Let us subsume the notation and the conditions of Proposition 3.7. Then, it holds that  $f(K) \leq 2f(r_K)$ .*

**Proof.** Notice that for type I cluster  $K$ ,  $f(K)$  consists of the cost of  $K$  and the cost of the path connecting it to the root  $r$  of  $T$  (to be precise, excluding  $r_K$  from this path, as we already count its cost in the cluster). Thus, the following holds

$$f(K) = c(K) + c(P_o(r_K)) \leq c(P(r_K)) + c(P_o(r_K)) \leq 2c(P(r_K)) = 2f(r_K),$$

where the first inequality is implied by Proposition 3.7, the second one from the fact that we added the cost of  $r_K$  to the right side, and the last inequality is by the definition of  $f$ .  $\blacktriangleleft$

Given the above, we can prove the main theorem of this section.

► **Theorem 3.9.** *For any MLA service function  $f$ , there exists a disjoint service function  $g$  that  $O(\sqrt{n})$ -approximates  $f$ . It can be found in time polynomial w.r.t. the MLA instance defining  $f$ .*

**Proof.** Let  $(T, c)$  be the MLA instance that defines  $f$ , and let  $U$  be the set of nodes in  $T$ . The idea is to prove that the partition  $P = \{P_1, P_2, \dots, P_k\}$  generated on  $T$  by Algorithm 2 induces a set function  $g(V) = \sum_{i \in [k]} f(P_i) \mathbb{1}\{P_i \cap V \neq \emptyset\}$  on all subsets  $V \subseteq U$  that is an  $O(\sqrt{n})$ -approximation to  $f$ . The function  $g$  is a disjoint service function by design.

For this purpose, we need to show that  $\max_{V \subseteq U} g(V)/f(V)$  is of order at most  $\sqrt{n}$ . Let us note that in our case,  $f(V)$  is just the cost of the minimal subtree connecting  $V$  to the root. Thus, for any subset  $V'$  of  $V$  it holds that  $f(V') \leq f(V)$ .

Let  $V \subseteq U$  be any subset of nodes and let  $P_{h,1}, \dots, P_{h,s}$  and  $P_{\ell,1}, P_{\ell,2}, \dots, P_{\ell,t}$  be the lists of all the heavy and light clusters that intersect  $V$ , respectively. We also denote the roots of the heavy clusters by  $v_{h,1}, \dots, v_{h,s}$ , respectively, and the set containing them by  $V_h$ . Similarly, we use the convention from Proposition 3.7 to define light cluster nodes. For  $P_{\ell,i}$ , we denote its root by  $r_{\ell,i}$ .

By Proposition 3.3, it holds that  $\sum_{i=1}^s f(P_{h,i}) \leq 8\sqrt{n} \cdot f(V_h)$ . Moreover, since  $V$  intersects all these heavy clusters, it either contains their roots or some nodes that are their descendants. Thus, the minimal subtree connecting  $V$  to the root  $r$  contains the minimal subtree connecting  $V_h$  to the root  $r$ . Hence,

$$\sum_{i=1}^s f(P_{h,i}) \leq 8\sqrt{n} \cdot f(V_h) \leq 8\sqrt{n} \cdot f(V). \quad (2)$$

Now, for each light cluster  $P_{\ell,i}$ , we notice that since  $V$  intersects it, the minimal tree connecting  $V$  to  $r$  contains the path from  $r_{\ell,i}$  to  $r$ . Thus,  $f(V) \geq f(r_{\ell,i})$  and by Proposition 3.3, we get that

$$f(P_{\ell,i}) \leq 2f(r_{\ell,i}) \leq 2f(V) \quad (3)$$

for each  $\ell \in [t]$ . Note that  $g(V) = \sum_{K \in P: V \cap K \neq \emptyset} f(K)$ . Combining inequalities 2 and 3, we obtain that

$$\begin{aligned} \frac{g(V)}{f(V)} &= \frac{\sum_{K \in P: V \cap K \neq \emptyset} f(K)}{f(V)} = \frac{\sum_{i=1}^s f(P_{h,i}) + \sum_{i=1}^t f(P_{\ell,i})}{f(V)} \leq \frac{8\sqrt{n} \cdot f(V) + \sum_{i=1}^t 2f(V)}{f(V)} \\ &\leq \frac{8\sqrt{n} \cdot f(V) + 2(\sqrt{n} + 1) \cdot f(V)}{f(V)} = 10\sqrt{n} + 2, \end{aligned}$$

with the last inequality implied by Proposition 3.6. This concludes the proof that  $g$  is an  $O(\sqrt{n})$ -approximation to  $f$ .

## 12:14 Universal Optimization for Non-Clairvoyant Subadditive Joint Replenishment

Finally, it is easy to notice that the algorithm runs in polynomial time. We can define a dynamic structure over the tree  $T$  that, for each node  $v$ , stores its subtree and path costs ( $c(T(v))$ ,  $c(P(v))$ ), together with the size  $|T(v)|$  of its subtree. Updates on such a structure take at most polynomial time in  $n$  (as we create a cluster, we go from the cluster root to the root of  $T$ , updating the data on all the nodes on the path, which is of length at most  $n$ ). With such a structure, checking whether a node is heavy or going through a path from  $r$  in search of a light cluster also takes at most linear time in  $n$ . ◀

Thus, by Lemma 1.1, we get Theorem 1.2. The result of Theorem 3.9 is tight:

► **Proposition 3.10.** *There exists a decreasing MLA instance  $T, c$  with  $n$  nodes, such that for every partition  $P_1, \dots, P_k$  of  $T$  for some  $k$ , there exists a non-empty set  $S \subseteq T$  such that*

$$\frac{\sum_{i \in [k]} f(P_i) \mathbb{1}\{S \cap P_i \neq \emptyset\}}{f(S)} = \Omega(\sqrt{n}).$$

**Proof.** Consider the tree  $T$  with a root  $r$  and  $n - 1$  children of  $r$  denoted by  $v_1, \dots, v_{n-1}$ . The cost  $c$  is such that  $c(r) = \sqrt{n}$ , while for all  $i \in [n - 1]$ ,  $c(v_i) = 1$ . Now, consider any partition  $P_1, \dots, P_k$ . If  $k > \sqrt{n}$ , then consider a set  $S$  that intersects each  $P_i$  exactly once. Thus,  $f(S) \leq \sqrt{n} + k \leq 2k$ , while  $\sum_{i \in [k]} f(P_i) \mathbb{1}\{S \cap P_i \neq \emptyset\} \geq k \cdot \sqrt{n}$ , which proves this case. Else ( $k \leq \sqrt{n}$ ), consider a set  $S$  that intersects  $P_i$  once if and only if  $|P_i| < \sqrt{n}/2$  (otherwise does not intersect at all). It holds that  $f(S) \leq 2\sqrt{n}$  because one has  $\sqrt{n}$  from  $r$  and  $\sqrt{n}$  intersections with  $P_i$ 's in the worst case, while

$$\sum_{i \in [k]} f(P_i) \mathbb{1}\{S \cap P_i \neq \emptyset\} \geq n - \sum_{i \in [k]} |P_i| \cdot \mathbb{1}\{S \cap P_i = \emptyset\} \geq n - k\sqrt{n}/2 \geq n/2,$$

which concludes the proof. ◀

## 4 Weighted Symmetric Subadditive Joint Replenishment

In this section, we study Weighted Symmetric Subadditive JRP. We have a set  $U$  of  $n$  request types with weights  $w(\{j\}) = w_j$  for each  $j \in U$ . Let  $f$  be the set function over  $U$ : In this setting, we have that the service cost of a set  $S$  only depends on the total weight of the elements belonging to  $S$ , as opposed to the identity of those elements. Formally,  $f(S) = f(w(S))$ , where function  $f$  is now intended as a monotone non-decreasing subadditive function of weights of a set with  $f(0) = 0$ , and for every two weights  $x, y$ , it holds that  $f(x + y) \leq f(x) + f(y)$ . For brevity, we call these functions *weighted symmetric subadditive*. Our goal is to show that for every weighted symmetric subadditive service function  $f$  on  $U$ , there exists a partition of  $U$  into sets  $S_1, \dots, S_k$  for some  $k$ , such that the disjoint service function  $g : U \rightarrow \mathbb{R}_{\geq 0}$  defined by this partition where  $g(S) = \sum_{i=1}^k f(S_i) \cdot \mathbb{1}[S \cap S_i \neq \emptyset]$  satisfies  $g(S) \leq O(\sqrt{n})f(S)$  for every  $S \subseteq U$ .

We begin, in Section 4.1, by analyzing a special case of unweighted symmetric subadditive service costs. Namely, where the weight of each element is 1, and thus,  $w(S) = |S|$ : These functions are simply referred to as *symmetric subadditive*. We achieve a tight  $\Theta(\sqrt{n})$ -stretch with a simple partitioning algorithm (partition into  $\sqrt{n}$  sets of size  $\sqrt{n}$  each), and this serves as a warm-up to the weighted symmetric subadditive case presented in Section 4.2, where we also achieve a tight  $\Theta(\sqrt{n})$ -stretch.

## 4.1 Symmetric Subadditive JRP

We first consider symmetric subadditive service functions. Observe that these functions are symmetric (i.e.,  $f(S) = f(S')$  for all sets  $S, S' \subseteq U$  such that  $|S| = |S'|$ ). For convenience, for a cardinality  $0 \leq s \leq |U|$ , we use  $f(s)$  as the value of sets of size  $s$ . We show that for symmetric subadditive  $f$ , one can construct a disjoint service function  $g$  that  $O(\sqrt{n})$ -approximates  $f$ . We then show that  $O(\sqrt{n})$  is tight even in the special case of  $f$  being a symmetric unweighted set cover function. This provides an alternative, simpler proof for the lower bound on USC of [35]. We first state the following simple but useful observation.

► **Observation 4.1.** *For all symmetric subadditive functions  $f : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ , and all  $y \geq x > 0$ , it holds that  $f(y)/f(x) \leq \lceil y/x \rceil$ .*

**Proof.** Let  $k = \lceil y/x \rceil$ . Then,  $f(y) \leq f(kx) \leq f(x) + f((k-1)x) \leq f(x) + \dots + f(x) = k \cdot f(x)$ . The first inequality is by monotonicity, and the second and third by subadditivity. ◀

► **Lemma 4.2.** *For every symmetric subadditive service function  $f$ , there exists a disjoint service function  $g$  that  $O(\sqrt{n})$ -approximates it.*

**Proof.** Let us consider an arbitrary symmetric subadditive service function  $f$  on request types  $U$ . Let  $g$  be the disjoint service function that induces an arbitrary partition of the elements of  $U$  into sets  $\{X_1, \dots, X_k\}$ , where  $k = \lceil \sqrt{n} \rceil$ , each of cardinality  $|X_i| \leq \lceil \sqrt{n} \rceil$  (such a partition always exists). We now bound the following fraction for every  $S \subseteq U$ :

$$\begin{aligned} \frac{\sum_{i \in [k]} f(X_i) \cdot \mathbb{1}\{S \cap X_i \neq \emptyset\}}{f(S)} &\leq \frac{\sum_{i \in [k]} f(\lceil \sqrt{n} \rceil) \cdot \mathbb{1}\{S \cap X_i \neq \emptyset\}}{f\left(\sum_{i \in [k]} \mathbb{1}\{S \cap X_i \neq \emptyset\}\right)} \\ &\leq \left( \sum_{i \in [k]} \mathbb{1}\{S \cap X_i \neq \emptyset\} \right) \cdot \left[ \frac{\lceil \sqrt{n} \rceil}{\sum_{i \in [k]} \mathbb{1}\{S \cap X_i \neq \emptyset\}} \right] \\ &\leq 2\lceil \sqrt{n} \rceil. \end{aligned}$$

The first inequality is because  $|X_i| \leq \lceil \sqrt{n} \rceil$  and from the fact that, since  $X_i$ 's are disjoint, the size of  $S$  is at least the number of non-empty intersections with sets  $X_i$ 's. The second inequality follows from Observation 4.1, and the third inequality follows since  $\lceil \frac{a}{b} \rceil \leq 2 \cdot \frac{a}{b}$ , for every  $\frac{a}{b} \geq \frac{1}{2}$ , and the denominator  $\sum_{i \in [k]} \mathbb{1}\{S \cap X_i \neq \emptyset\} \leq \sqrt{n} + 1$ . ◀

Thus, by Lemma 1.1 and Lemma 4.2, the following holds:

► **Theorem 4.3.** *There exists a deterministic  $O(\sqrt{n})$ -competitive algorithm for the Non-Clairvoyant Symmetric Subadditive Joint Replenishment problem.*

We complement the above result by giving a tight instance:

► **Theorem 4.4.** *There exists a symmetric subadditive service function such that every disjoint service function is an  $\Omega(\sqrt{n})$ -approximation of it.*

**Proof.** Let  $U$  be the set of request types. For simplicity of the proof, we assume that  $n = |U|$  has an integer square root. Let us consider the service function  $f(S) = \left\lceil \frac{|S|}{\sqrt{n}} \right\rceil$ , which is symmetric and subadditive, let  $g$  be any disjoint service function, let  $\mathcal{S}$  be the collection of disjoint sets  $X_i$ 's that  $g$  generates, and let  $k$  be the number of parts in the partition  $\mathcal{S}$ .

## 12:16 Universal Optimization for Non-Clairvoyant Subadditive Joint Replenishment

Consider some  $X \subseteq U$  that intersects each  $X_i$  exactly once. We now analyze the cost of this induced partition on  $X$ :

$$\frac{\sum_{i=1}^k f(X_i)}{f(X)} = \frac{\sum_{i=1}^k \left\lceil \frac{|X_i|}{\sqrt{n}} \right\rceil}{\left\lceil \frac{k}{\sqrt{n}} \right\rceil} \geq \frac{\max\{k, \sqrt{n}\}}{\left\lceil \frac{k}{\sqrt{n}} \right\rceil},$$

where the inequality holds since it is a sum of  $k$  terms where each is at least 1, and since the sets  $X_1, \dots, X_k$  cover  $U$ , thus  $\sum_{i=1}^k |X_i| = n$ .

Now, if  $k \leq \sqrt{n}$  then

$$\frac{\max\{k, \sqrt{n}\}}{\left\lceil \frac{k}{\sqrt{n}} \right\rceil} = \sqrt{n}.$$

Otherwise,  $\frac{k}{\sqrt{n}} > 1$ , thus  $\left\lceil \frac{k}{\sqrt{n}} \right\rceil \leq 2\frac{k}{\sqrt{n}}$ , which implies that

$$\frac{\max\{k, \sqrt{n}\}}{\left\lceil \frac{k}{\sqrt{n}} \right\rceil} \geq \frac{k}{2k/\sqrt{n}} = \frac{\sqrt{n}}{2},$$

which concludes the proof.  $\blacktriangleleft$

### 4.2 Weighted Symmetric Subadditive JRP

We now relax the assumption of  $w(S) = |S|$  and provide a  $O(\sqrt{n})$ -approximation for every weighted subadditive function. We begin with some facts about weighted subadditive and symmetric concave functions. Every symmetric concave function is the pointwise infimum of a set of affine functions, and can be approximated by a set of affine functions with exponentially decreasing slopes. The next lemma combines this fact with the fact that every weighted subadditive function can be approximated by a symmetric concave function.

► **Lemma 4.5.** *Let  $g : \{0, 1, \dots, W\} \rightarrow \mathbb{R}_{\geq 0}$  be a monotone non-decreasing subadditive function. Then, there exists a finite set of affine functions  $\{g_1, \dots, g_p\}$  for some  $p \leq \log(W)$  where  $g_i(x) = \sigma_i + x \cdot \delta_i$  such that  $\sigma_{i+1} > 2\sigma_i$  and  $\delta_{i+1} < \delta_i/2$  for every  $i < p$ , and the function  $\hat{g}$  defined by  $\hat{g}(x) = \min_i g_i(x)$  satisfies that for every  $x \in \{0, \dots, W\}$ , it holds that:*

$$g(x) \leq \hat{g}(x) \leq 8g(x).$$

**Proof.** By [30], we know that there exists a concave function  $g' : \{0, \dots, W\} \rightarrow \mathbb{R}_{\geq 0}$  that approximates  $g$  within a factor of 2. Now, for every  $i = 2, \dots, \lceil \log(W) \rceil$  consider the affine function  $g'_i : \{0, \dots, W\} \rightarrow \mathbb{R}_{\geq 0}$  that interpolates between  $(2^{i-1}, g'(2^{i-1}))$  and  $(2^i, g'(2^i))$ , and  $g'_1(x)$  that interpolates between  $(0, g'(0))$  and  $(1, g'(1))$ . It holds that for every  $x \in \{0, \dots, W\}$  then

$$\frac{g'(x)}{2} \leq \min_{i=1, \dots, p} g'_i(x) \leq g'(x),$$

where the first inequality holds since

$$\begin{aligned} g'(x) &\leq g'(2^{\lceil \log(x) \rceil}) \leq 2g'(2^{\lfloor \log(x) \rfloor}) \\ &\leq 2g'_{2^{\lfloor \log(x) \rfloor}}(2^{\lfloor \log(x) \rfloor}) = 2 \min_{i=1, \dots, p} g'_i(2^{\lfloor \log(x) \rfloor}) \leq 2 \min_{i=1, \dots, p} g'_i(x), \end{aligned}$$

and the second inequality holds by concavity of  $g'$ . In [32], they present an algorithm that reduces the set of affine functions such that the coefficients and slopes satisfy the conditions of the lemma while losing a factor of 2, which, if applied to the set of affine functions  $2g'_i$ , concludes the proof.  $\blacktriangleleft$

Henceforth, we will denote by  $W = w(U)$ , and assume that  $f$  is defined on  $\{0, \dots, W\}$ , and is a pointwise infimum of  $p$  affine functions  $g_1, \dots, g_p$  where  $g_i(x) = \sigma_i + x \cdot \delta_i$  and the  $\sigma_i$ s and  $\delta_i$ s satisfy the properties stated in the lemma. Proving the theorem for  $f$  that satisfies the condition proves the same (with additional loss of a factor of 8) for general symmetric subadditive functions.

The following lemma will be useful to lower bound  $f(w(S))$  using the largest weight in  $S$ .

► **Lemma 4.6.** *For every  $k \in \{2, \dots, p\}$ , if  $x \geq \frac{\sigma_k}{\delta_{k-1}}$ , then  $f(x) \geq \sigma_k$ .*

**Proof.** Recall that  $f(x) = \min_{1 \leq i \leq p} \sigma_i + x\delta_i$ . For  $i < k$ , we have  $\sigma_i + x\delta_i \geq x\delta_{k-1} \geq \sigma_k$ . For  $i \geq k$ , we have  $\sigma_i + x\delta_i \geq \sigma_k$ . Thus,  $f(x) \geq \sigma_k$ . ◀

Henceforth, for brevity, we write  $f(S)$  to mean  $f(w(S))$ , for an arbitrary set  $S$ . In the following, we frequently use the fact that for any set  $H$ ,  $f(H) = \min_{1 \leq i' \leq p} \sigma_{i'} + w(S)\delta_{i'} \leq \sigma_i + w(H)\delta_i$  for every  $i$ .

**High-Level Overview.** Let  $S$  be a set chosen by an adversary, unknown to us. Suppose that  $f(S) = \min_{1 \leq i \leq p} \sigma_i + w(S)\delta_i = \sigma_\ell + w(S)\delta_\ell$ . The idea is to construct a partition such that some of the parts that intersect  $S$  can be charged to  $\sigma_\ell$ , and the remaining parts that intersect  $S$  can be charged to  $w(S)\delta_\ell$ . Towards this end, we first classify each type  $j$  as follows. We say that type  $j$  is *eligible* for class  $2 \leq k \leq p$  if  $w_j \geq \frac{\sigma_k}{\delta_{k-1}}$ . All types are eligible for class 1. Define the *class* of type  $j$  to be the largest class it is eligible for and  $X_k$  to be the set of class- $k$  types.

Next, we partition  $X_k$  into heavy and light types. The light part  $Z_k$  contains all types  $j \in X_k$  with  $w_j\delta_k \leq \sigma_k/\sqrt{n}$ . Since  $Z_k$  is light,  $f(Z_k) \leq \sigma_k + w(Z_k)\delta_k \leq O(\sqrt{n})\sigma_k$ . Also, if  $S \cap X_k \neq \emptyset$ , then Lemma 4.6 implies that  $f(S) \geq \sigma_k$ . We can then use the fact that  $\sigma_k$ 's are geometric to show that the total value of the parts  $Z_k$  that intersect  $S$  is at most  $O(\sqrt{n})f(S)$ .

Now, consider the heavy types in  $X_k$ , i.e. those types  $j$  with  $w_j\delta_k > \sigma_k/\sqrt{n}$ . We further partition these types according to their weights in powers of 2. Let  $R_{k,i} = \{j \in X_k \setminus Z_k : w_j \in [2^i, 2^{i+1})\}$ . For each weight class  $i$ , we greedily partition  $R_{k,i}$  into as many parts of size  $\lceil \sqrt{n} \rceil$  as we can. This produces a collection  $F_{k,i}$  of parts of size  $\lceil \sqrt{n} \rceil$  and at most one leftover part  $G_{k,i}$  of size less than  $\sqrt{n}$ . We say that a part is *nice* if it belongs to  $F_{k,i}$  and the part  $G_{k,i}$  a *leftover* part.

Observe that there are at most  $\lceil \sqrt{n} \rceil$  nice parts, each of size at most  $\lceil \sqrt{n} \rceil$  and contains types of roughly the same weight. Thus, we can use a similar argument as in the unweighted case to show that the total value of the nice parts that intersect  $S$  is at most  $O(\sqrt{n})f(S)$ . For the leftover parts, we charge the parts  $G_{k,i}$  that intersect  $S$  with  $k < \ell$  to  $w(S)\delta_\ell$  and those with  $k \geq \ell$  to  $\sigma_\ell$ .

■ **Algorithm 3** Weighted Symmetric Subadditive Partitioning Algorithm.

- 
- 1: **for**  $k = 1$  to  $p$  **do**
  - 2:   Create a part  $Z_k = \{j \in X_k : w_j\delta_k \leq \sigma_k/\sqrt{n}\}$
  - 3:   Let  $R_{k,i} = \{j \in X_k \setminus Z_k : w_j \in [2^i, 2^{i+1})\}$
  - 4:   **for each**  $i$  **do**
  - 5:     Greedly partition  $R_{k,i}$  into as many sets of size exactly  $\lceil \sqrt{n} \rceil$  as possible
  - 6:     Let  $F_{k,i}$  denote the sets of size of size  $\lceil \sqrt{n} \rceil$
  - 7:     Let  $G_{k,i}$  denote the remaining set of size less than  $\sqrt{n}$ , if it exists
  - 8:     Create a part for each set in  $F_{k,i}$  and a part for the set  $G_{k,i}$
-

We now give the detailed analysis below.

► **Theorem 4.7.** *For any weighted symmetric subadditive service function  $f$ , there exists a disjoint service function  $g$  that  $O(\sqrt{n})$ -approximates  $f$ . It can be found in time polynomial w.r.t. the weights defining  $f$ .*

**Proof.** Let  $S$  be an arbitrary set and suppose  $f(S) = \min_{1 \leq i \leq p} \sigma_i + w(S)\delta_i = \sigma_\ell + w(S)\delta_\ell$ . We now decompose  $w(S)$  using the partition produced by our algorithm. In particular, we have

$$f(S) = \sigma_\ell + \left( \sum_k w(Z_k \cap S) + \sum_{k,i} \sum_{T \in F'_{k,i}} w(T \cap S) + \sum_{k,i} w(G_{k,i} \cap S) \right) \cdot \delta_\ell.$$

Define  $F'_{k,i}$  as the subset of parts in  $F_{k,i}$  that intersects with  $S$ . We now show that the algorithm pays at most  $O(\sqrt{n})f(S)$ . In other words, we will prove that the total value of the parts that intersect  $S$  are upper bounded as follows:

$$\sum_{k: Z_k \cap S \neq \emptyset} f(Z_k) + \sum_{k,i} \sum_{T \in F'_{k,i}} f(T) + \sum_{k,i: G_{k,i} \cap S \neq \emptyset} f(G_{k,i}) \leq O(\sqrt{n})f(S).$$

We begin by bounding  $\sum_{k: Z_k \cap S \neq \emptyset} f(Z_k)$ . Let  $k_{\max}$  be the largest  $k$  such that  $Z_k \cap S \neq \emptyset$ . (If none exists, then we do not need to bound this term.) We have that  $w(S) \cdot \delta_{k_{\max}-1} \geq w(Z_k \cap S) \cdot \delta_{k_{\max}-1} \geq \sigma_{k_{\max}}$ . Thus, Lemma 4.6 implies that  $f(S) \geq \sigma_{k_{\max}}$ . On the other hand,

$$\sum_{k: Z_k \cap S \neq \emptyset} f(Z_k) \leq \sum_{k: Z_k \cap S \neq \emptyset} O(\sqrt{n})\sigma_k \leq O(\sqrt{n})\sigma_{k_{\max}} \leq O(\sqrt{n})f(S).$$

where the first inequality follows directly from the definition of  $Z_k$  in line 2 of Algorithm 3 and since there are at most  $n$  elements in  $Z_k$ , the second inequality is since the  $\sigma_k$ 's are geometrically increasing.

Next, we bound  $\sum_{k,i} \sum_{T \in F'_{k,i}} f(T)$ . Since every set  $T \in F'_{k,i}$  has size  $\lceil \sqrt{n} \rceil$ , we have  $\sum_{k,i} |F'_{k,i}| \leq \sqrt{n}$ . Moreover, every  $j \in T$  has  $w_j \in [2^i, 2^{i+1})$ , so  $w(T) \leq O(\sqrt{n})w(T \cap S)$ . Thus, we have

$$\begin{aligned} \sum_{k,i} \sum_{T \in F'_{k,i}} f(T) &\leq \sum_{k,i} \sum_{T \in F'_{k,i}} \sigma_\ell + w(T)\delta_\ell \\ &\leq \sum_{k,i} |F'_{k,i}| \sigma_\ell + O(\sqrt{n}) \sum_{k,i} \sum_{T \in F'_{k,i}} w(T \cap S)\delta_\ell \\ &\leq O(\sqrt{n}) \left( \sigma_\ell + \sum_{k,i} \sum_{T \in F'_{k,i}} w(T \cap S)\delta_\ell \right) \leq O(\sqrt{n})f(S). \end{aligned}$$

where the last inequality follows from the fact that all  $T \in F'_{k,i}$  are disjoint so we have that  $w(S) \geq \sum_{k,i} \sum_{T \in F'_{k,i}} w(T \cap S)$ .

We now turn to bounding  $\sum_{k,i: G_{k,i} \cap S \neq \emptyset} f(G_{k,i})$ . Consider a set  $G_{k,i}$  that intersects  $S$  for  $\ell \leq k \leq p$ . Since  $G_{k,i}$  is a subset of  $X_k \setminus Z_k$  and is at most of size  $\sqrt{n}$ , we have that

$$f(G_{k,i}) \leq \sigma_k + w(G_{k,i})\delta_k \leq O(\sqrt{n})w(G_{k,i} \cap S)\delta_k.$$



Since  $\delta_k \leq \delta_\ell$ , we get that

$$\sum_{k \geq \ell} \sum_{i: G_{k,i} \cap S \neq \emptyset} f(G_{k,i}) \leq \sum_{k \geq \ell} \sum_{i: G_{k,i} \cap S \neq \emptyset} O(\sqrt{n})w(G_{k,i} \cap S)\delta_\ell \leq O(\sqrt{n})f(S).$$

Finally, when  $\ell = 1$ , the argument is complete. Let us now consider the case when  $\ell > 1$ . Consider a set  $G_{k,i}$  that intersects  $S$  for  $1 \leq k < \ell$ . We have that  $f(G_{k,i}) \leq \sigma_k + w(G_{k,i})\delta_k \leq O(\sqrt{n})2^{i+1}\delta_k$ . Moreover, since every  $j \in X_k$  has  $w_j\delta_k < \sigma_{k+1}$ , we have that

$$\sum_{k < \ell} \sum_{i: G_{k,i} \cap S \neq \emptyset} f(G_{k,i}) \leq O(\sqrt{n}) \sum_{k < \ell} \sigma_{k+1} \leq O(\sqrt{n})\sigma_\ell \leq O(\sqrt{n})f(S).$$

Finally, it is not hard to see that, by design, Algorithm 3 can be implemented in polynomial time in the logarithm of the total weight,  $\log(w(U))$ . This concludes the proof. ◀

Thus, by Lemma 1.1, we get Theorem 1.3.

## 5 Tight Instances against Previous Algorithms

### 5.1 An $\Omega(\sqrt{n \log n})$ Tight Instance for the Algorithm of [35]

► **Proposition 5.1.** *There exists a weighted set cover instance for which the Universal Set Cover algorithm of [35] has stretch  $\Omega(\sqrt{n \log n})$ .*

**Proof.** The algorithm of [35] works as follows: while the set  $U$  of elements  $e$  for which  $f(e)$  is undefined is non-empty, pick the set  $S$  that minimizes  $\frac{c(S)}{\sqrt{|S \cap U|}}$  and for all  $e \in S \cap U$ , define  $f(e) = S$ .

The high-level idea is that [35]'s analysis uses the Cauchy-Schwarz inequality and the tight instance is created by looking at when the Cauchy-Schwarz inequality is tight.

Consider the following set system where we have sets  $S, S_1, \dots, S_k$  for some  $k$  that we will choose later. The set  $S$  contains  $k$  elements and set  $S_i$  contains  $\lfloor \frac{k}{k-(i-1)} \rfloor$  elements. The sets also satisfy that  $|S \cap S_i| = 1$  and  $S_i \cap S_j = \emptyset$  for  $1 \leq i < j \leq k$ . Moreover, the sets  $S_i$  form a partition of all the  $n$  elements. The costs of the sets are:  $c(S) = 1$ ,  $c(S_i) = \frac{\sqrt{|S_i|}}{\sqrt{k-(i-1)}}$ .

We now claim that in the  $i$ -th iteration, the algorithm chooses  $S_i$ . First observe that for  $1 \leq i < j \leq k$ , we have

$$\frac{c(S_i)}{\sqrt{|S_i|}} < \frac{c(S_j)}{\sqrt{|S_j|}}.$$

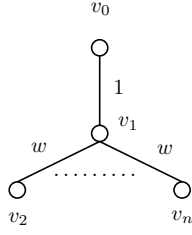
Thus, it suffices to show that in each iteration  $i$ , the algorithm chooses  $S_i$  over  $S$ . We do this by induction on  $i$ . When  $i = 1$ , we have that

$$\frac{c(S_1)}{\sqrt{|S_1|}} = \frac{1}{\sqrt{k}} = \frac{c(S)}{\sqrt{|S|}}.$$

Now consider  $i > 1$ . By induction, we have that  $|S \cap U| = k - (i - 1)$  and  $S_i \cap U = S_i$  (the latter is because the only set that intersects  $S_i$  is  $S$ ). Thus, we also have

$$\frac{c(S_i)}{\sqrt{|S_i|}} = \frac{1}{\sqrt{k - (i - 1)}} = \frac{c(S)}{\sqrt{|S \cap U|}}.$$

We conclude that in each iteration  $i$ , the algorithm chooses  $S_i$ .



■ **Figure 2** Tight instance for [44], where  $w = \frac{\sqrt{n \log n - 1}}{n-1}$ .

Thus, the competitive ratio of the algorithm is at least  $(\sum_{i=1}^k c(S_i))/c(S) = \sum_{i=1}^k c(S_i)$  since  $c(S) = 1$ . We have that

$$\sum_{i=1}^k c(S_i) = \sum_{i=1}^k \frac{\sqrt{\left\lfloor \frac{k}{k-(i-1)} \right\rfloor}}{\sqrt{k-(i-1)}} = \Omega(\sqrt{k} \log k). \quad (4)$$

It now remains to maximize  $k$ . The constraint on  $k$  is that  $\sum_{i=1}^k |S_i| = n$  since  $S_i$ s are disjoint. Now,  $\sum_{i=1}^k |S_i| = \sum_{i=1}^k \left\lfloor \frac{k}{k-(i-1)} \right\rfloor = \Theta(k \log k)$ . Thus, setting  $k = \Theta(n/\log n)$  satisfies the constraint on  $k$ . Plugging this into (4) yields the claim. ◀

## 5.2 An $\Omega(\sqrt{n \log n})$ Tight Instance for the Algorithm of [44]

We complement the  $O(\sqrt{n})$ -stretch achieved by Algorithm 2 and Algorithm 3 with a JRP instance such that the algorithm of [44] (Algorithm 2) must suffer a stretch of at least  $\Omega(\sqrt{n \log n})$ . Note that the instance we present in Figure 2 is both an MLA instance and a weighted concave one. This shows that for the specific case of MLA and weighted concave functions, not only is our algorithm optimal, but also that Touitou's algorithm cannot achieve the same guarantee. At a high level, whenever Touitou's algorithm decides to serve some requests, it issues up to two services (lines 9 and 12). One of them serves a subset of requests  $R$  for which delay and service costs are the same. At the same time, a second service with a budget of up to  $\sqrt{n \log n} \cdot c(R)$  can be issued to serve some pending requests in advance. The following example is one where the optimal algorithm rarely issues this second service.

► **Proposition 5.2.** *There exists an instance for which the algorithm of [44] has stretch  $\Omega(\sqrt{n \log n})$ . Moreover, this is an MLA and a weighted concave instance.*

**Proof.** Let us consider the JRP tree  $T$  in Figure 2, where  $w = \frac{\sqrt{n \log n - 1}}{n-1}$  and the delay cost functions on the nodes read

$$d_i(t) = \begin{cases} 2t, & \text{if } i = 1 \\ \varepsilon t, & \text{if } i \geq 2 \end{cases},$$

for  $\varepsilon \ll w$  to be set later. In particular, at each time step, there are  $n$  requests arriving on tree  $T$ , one per node.

Let us first observe that the optimum algorithm only serves the requests at  $v_1$ , paying a service cost of 1 at each time step. Moreover, it serves requests arriving at any  $v_i$  with  $i \geq 2$  once  $\varepsilon t = w$ , i.e., every  $w/\varepsilon$  time steps, and pays  $(n-1)w + 1 = \sqrt{n \log n}$ . Thus, letting  $\tau$  be the length of the requests sequence, the overall optimal cost is  $\text{OPT}(\tau) = \tau + \frac{\varepsilon \tau}{w} \sqrt{n \log n} \leq 2\tau$ , by setting  $\varepsilon = w/n$ .

Algorithm 2 in [44] (whose cost is referred to as ALG from now on) serves a request arriving at  $v_1$  (line 9) as soon as its accumulated delay equals its service cost (this is when the UPONCRITICAL event occurs). Once a request at  $v_1$  arrives, the algorithm waits until the time elapsed  $t$  is such that  $2t = 1$  to serve it. That is, when the  $j$ -th request located at  $v_1$  arrives, the algorithm serves it at time  $t_j = j + \frac{1}{2}$ . Right after, it issues a second service (line 12) to serve all other requests at  $v_2, \dots, v_n$ . Overall, the algorithm pays  $\text{ALG}(\tau) = \tau \cdot (1 + (n-1)w) = \tau\sqrt{n \log n}$ .

Hence,

$$\frac{\text{ALG}(\tau)}{\text{OPT}(\tau)} \geq \frac{\sqrt{n \log n}}{2},$$

for all  $\tau \geq 1$ . To conclude, the fact that the instance in Figure 2 is an MLA one comes directly from the fact that it is a depth 2 tree. Moreover, observe that no matter how we choose  $S \subseteq V$ ,  $f(S) = f(w(S))$ , and thus the instance in Figure 2 is also a weighted concave instance. ◀

---

## References


- 1 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, volume 81 of *LIPICs*, pages 1:1–1:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.APPROX-RANDOM.2017.1.
- 2 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1051–1061. SIAM, 2017. doi:10.1137/1.9781611974782.67.
- 3 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay – Clairvoyance is not required. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPICs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ESA.2020.8.
- 4 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. *Theory Comput. Syst.*, 64(4):572–592, 2020. doi:10.1007/s00224-019-09963-7.
- 5 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalya Panigrahi. Online service with delay. *ACM Trans. Algorithms*, 17(3):23:1–23:31, 2021. doi:10.1145/3459925.
- 6 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 301–320. SIAM, 2021. doi:10.1137/1.9781611976465.20.
- 7 Yossi Azar and Noam Touitou. General framework for metric optimization problems with delay or with deadlines. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 60–71. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00013.
- 8 Yossi Azar and Noam Touitou. Beyond tree embeddings – a deterministic framework for network design with deadlines or delay. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1368–1379. IEEE, 2020. doi:10.1109/FOCS46700.2020.00129.
- 9 Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035. SIAM, 2012.
- 10 Kshipra Bhawalkar and Tim Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *SODA*, pages 700–709. SIAM, 2011.

- 11 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Kim Thang Nguyen, and Pavel Veselý. Online algorithms for multilevel aggregation. *Oper. Res.*, 68(1):214–232, 2020. doi:10.1287/opre.2019.1847.
- 12 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, Marek Chrobak, Christoph Dürr, Lukáš Folwarczný, Lukasz Jez, Jirí Sgall, Kim Thang Nguyen, and Pavel Veselý. New results on multilevel aggregation. *Theor. Comput. Sci.*, 861:133–143, 2021. doi:10.1016/j.tcs.2021.02.016.
- 13 Marcin Bienkowski, Martin Böhm, Jaroslaw Byrka, and Jan Marcinkowski. Online facility location with linear delay. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPICs*, pages 45:1–45:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.APPROX/RANDOM.2022.45.
- 14 Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, Dorian Nogneng, and Jirí Sgall. Better approximation bounds for the joint replenishment problem. 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 42–54. SIAM, 2014. doi:10.1137/1.9781611973402.4.
- 15 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In *Approximation and Online Algorithms – Proceedings of the 16th International Workshop, WAOA 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11312 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2018. doi:10.1007/978-3-030-04693-4\_4.
- 16 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. A match in time saves nine: Deterministic online matching with delays. In *Approximation and Online Algorithms – Proceedings of the 15th International Workshop, WAOA 2017*, volume 10787 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2017. doi:10.1007/978-3-319-89441-6\_11.
- 17 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. Online service with delay on a line. In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity – 25th International Colloquium, SIROCCO 2018, Ma’ale HaHamisha, Israel, June 18-21, 2018, Revised Selected Papers*, volume 11085 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 2018. doi:10.1007/978-3-030-01325-7\_22.
- 18 Carlos Fisch Brito, Elias Koutsoupias, and Shailesh Vaya. Competitive analysis of organization networks or multicast acknowledgment: How much to wait? *Algorithmica*, 64(4):584–605, 2012. doi:10.1007/s00453-011-9567-5.
- 19 Niv Buchbinder, Moran Feldman, Joseph (Seffi) Naor, and Ohad Talmon.  $O(\text{depth})$ -competitive algorithm for online multi-level aggregation. 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 1235–1244. SIAM, 2017. doi:10.1137/1.9781611974782.80.
- 20 Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Algorithms – ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, volume 4698 of *Lecture Notes in Computer Science*, pages 253–264. Springer, 2007. doi:10.1007/978-3-540-75520-3\_24.
- 21 Niv Buchbinder, Tracy Kimbrel, Retsef Levi, Konstantin Makarychev, and Maxim Sviridenko. Online make-to-order joint replenishment model: primal dual competitive algorithms. In Shang-Hua Teng, editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 952–961. SIAM, 2008. URL: <http://dl.acm.org/citation.cfm?id=1347082.1347186>.
- 22 Rodrigo A. Carrasco, Kirk Pruhs, Cliff Stein, and José Verschae. The online set aggregation problem. In Michael A. Bender, Martin Farach-Colton, and Miguel A. Mosteiro, editors, *LATIN 2018: Theoretical Informatics – 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings*, volume 10807 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2018. doi:10.1007/978-3-319-77404-6\_19.

- 23 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.40.
- 24 Lindsey Deryckere and Seeun William Umboh. Online matching with set and concave delays. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPICs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.APPROX/RANDOM.2023.17.
- 25 Shahar Dobzinski. Two randomized mechanisms for combinatorial auctions. In *APPROX-RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2007.
- 26 Shahar Dobzinski, Uriel Feige, and Michal Feldman. Are gross substitutes a substitute for submodular valuations? In *EC*, pages 390–408. ACM, 2021.
- 27 Daniel R. Dooly, Sally A. Goldman, and Stephen D. Scott. On-line analysis of the TCP acknowledgment delay problem. *J. ACM*, 48(2):243–273, 2001. doi:10.1145/375827.375843.
- 28 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 333–344. ACM, 2016. doi:10.1145/2897518.2897557.
- 29 Yuval Emek, Yaacov Shapiro, and Yuyi Wang. Minimum cost perfect matching with delays for two sources. *Theor. Comput. Sci.*, 754:122–129, 2019. doi:10.1016/j.tcs.2018.07.004.
- 30 Tomer Ezra, Michal Feldman, Tim Roughgarden, and Warut Suksompong. Pricing multi-unit markets. *ACM Trans. Economics and Comput.*, 7(4):20:1–20:29, 2020.
- 31 Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544. SIAM, 2009.
- 32 Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation for the single sink edge installation problem. *SIAM J. Comput.*, 38(6):2426–2442, 2009. doi:10.1137/050643635.
- 33 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Caching with time windows. 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 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.
- 34 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. A hitting set relaxation for  $k$ -server and an extension to time-windows. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 504–515. IEEE, 2021. doi:10.1109/FOCS52979.2021.00057.
- 35 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for tsp, steiner tree, and set cover. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 386–395. ACM, 2005. doi:10.1145/1060590.1060649.
- 36 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic TCP acknowledgement and other stories about  $e/(e-1)$ . 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 502–509. ACM, 2001. doi:10.1145/380752.380845.
- 37 Alexander S. Kelso and Vincent P. Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica*, 50(6):1483–1504, 1982.
- 38 Predrag Krnetić, Darya Melnyk, Yuyi Wang, and Roger Wattenhofer. The  $k$ -server problem with delays on the uniform metric space. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPICs*, pages 61:1–61:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ISAAC.2020.61.

- 39 Ngoc Mai Le, Seeun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1594–1610. SIAM, 2023. doi:10.1137/1.9781611977554.ch59.
- 40 Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games Econ. Behav.*, 55(2):270–296, 2006.
- 41 Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Impatient online matching. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC 2018*, volume 123 of *LIPICs*, pages 62:1–62:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ISAAC.2018.62.
- 42 Jeremy McMahan. A d-competitive algorithm for the multilevel aggregation problem with deadlines. *CoRR*, abs/2108.04422, 2021. arXiv:2108.04422.
- 43 Noam Touitou. Nearly-tight lower bounds for set cover and network design with deadlines/delay. 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 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ISAAC.2021.53.
- 44 Noam Touitou. Frameworks for nonclairvoyant network design with deadlines or delay. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, page to appear, 2023.
- 45 Noam Touitou. Improved and deterministic online service with deadlines or delay. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 761–774. ACM, 2023. doi:10.1145/3564246.3585107.

# The Average-Value Allocation Problem

Kshipra Bhawalkar  

Google Research, Mountain View, USA

Zhe Feng  

Google Research, Mountain View, USA

Anupam Gupta  

NYU & Google Research, New York City & Mountain View, USA

Aranyak Mehta  

Google Research, Mountain View, USA

David Wajc<sup>1</sup>  

Technion, Haifa, Israel

Di Wang  

Google Research, Mountain View, USA

---

## Abstract

We initiate the study of centralized algorithms for welfare-maximizing allocation of goods to buyers subject to *average-value constraints*. We show that this problem is NP-hard to approximate beyond a factor of  $\frac{e}{e-1}$ , and provide a  $\frac{4e}{e-1}$ -approximate offline algorithm. For the online setting, we show that no non-trivial approximations are achievable under adversarial arrivals. Under i.i.d. arrivals, we present a polytime online algorithm that provides a constant approximation of the optimal (computationally-unbounded) online algorithm. In contrast, we show that no constant approximation of the ex-post optimum is achievable by an online algorithm.

**2012 ACM Subject Classification** Theory of computation → Approximation algorithms analysis; Theory of computation → Online algorithms

**Keywords and phrases** Resource allocation, return-on-spend constraint, approximation algorithm, online algorithm

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.13

**Category** APPROX

**Related Version** *Full Version*: <http://arxiv.org/abs/2407.10401> [9]

**Funding** *Anupam Gupta*: Supported in part by NSF awards CCF-1955785 and CCF-2006953.

*David Wajc*: Supported in part by a Taub Family Foundation “Leader in Science and Technology” fellowship. Work done while the author was visiting Google Research.

## 1 Introduction

Allocating goods to buyers so as to maximize social welfare is one of the most central problems in economics. This problem, even under linear utilities, is complicated by buyers’ various constraints and the manner in which items are revealed.

In this work we introduce the *average-value allocation* problem (AVA). Here, we wish to maximize social welfare (total value of allocated items), while guaranteeing for each buyer  $j$  an *average* value of allocated items of at least  $\rho_j$ . Formally, if the value of item  $i$  for buyer  $j$  is  $v_{ij}$ , and  $x_{ij} \in \{0, 1\}$  indicates whether item  $i$  is allocated to buyer  $j$ , we wish to maximize the social welfare  $\sum_{i,j} v_{ij} x_{ij}$ , subject to each item being allocated to at most one buyer (i.e.,  $\sum_j x_{ij} \leq 1$ ), and to the “average value” constraint:

---

<sup>1</sup> corresponding author



## 13:2 The Average-Value Allocation Problem

$$\forall j, \quad \sum_i v_{ij} x_{ij} \geq \rho_j \cdot \left( \sum_i x_{ij} \right). \quad (1.1)$$

Average-value constraints arise naturally in numerous situations. E.g., consider settings when goods are to be distributed among “buyers”, and the (fixed) cost of distributing, receiving, or deploying each such good allocated is borne by the recipient. Each buyer wants their average value for their goods to be at least some parameter  $\rho_j$ . This parameter  $\rho_j$  allows to convert between units, and so this fixed cost for each buyer can be in money, time, labor, or any other unit. So, for example, for allocation and distribution of donations to a charitable organization, a certain value-per-item is required to justify the time contributed by volunteers, or the money spent by government in the form of subsidies. In other words, the amount of “benefit” per task allocated to an individual  $j$  should be above the threshold  $\rho_j$ , so that even if some of the tasks are individually less rewarding (i.e., they have benefit less than  $\rho_j$ , the total amount of happiness they get overall justifies their workload.

In addition to this average-value constraint on the allocation, we may also consider side-constraints (such as the well-studied budget constraints), but for now we defer their discussion and focus on on the novel constraint (1.1). At first glance, the AVA problem may seem similar to other packing problems in the literature, but there is a salient difference – it is not a packing problem at all! Indeed, if buyer  $i$  gets some subset  $S_i = \{j \mid x_{ij} = 1\}$  of items in some feasible allocation, it is possible that a subset  $S' \subseteq S_i$  of this allocation is no longer feasible, since its average value may be lower. Given that this packing (subset-closedness) property is crucial to many previous results on allocation problems, their techniques do not apply. Hence, we have to examine this problem afresh, and we ask: *how well can the average-value allocation be approximated?* We investigate this question, both in the offline and online settings.

### 1.1 Our Results and Techniques

Recall that the AVA problem seeks to maximize the social welfare  $\sum_{ij} v_{ij} x_{ij}$  subject to each item going to at most one buyer, and also the novel average-value constraint (1.1) above. Our first result rules out polynomial-time exact algorithms for AVA in an offline setting, or even a PTAS, showing that this problem is as hard to approximate as the MAX-COVERAGE problem:

► **Theorem 1 (Hardness of AVA).** *For any constant  $\varepsilon > 0$ , the AVA problem is NP-hard to  $(\frac{e}{e-1} - \varepsilon)$ -approximate.*

We then turn our attention to positive results, and give the following positive result for the problem:

► **Theorem 2 (Offline AVA).** *There exists a randomized polynomial-time algorithm for the AVA problem which achieves an approximation factor of  $\frac{4e}{e-1}$ .*

To prove Theorem 2, we would like to draw on techniques used for traditional packing problems, but the non-traditional nature of this problem means we need to investigate its structure carefully. A key property we prove and leverage throughout is the existence of approximately-optimal solutions of a very special kind: each buyer gets a collection of “bundles”, where a bundle for buyer  $j$  consists of a single item  $i$  with positive  $v_{ij} - \rho_j$  (i.e., contributing positively to the average-value constraint (1.1)) and some number of items  $i$  with



negative  $v_{ij} - \rho_j$ , such that they together satisfy the AVA constraint. Given this structure we can focus on partitioning items among bundles, and allocating bundles to buyers. Note that this partitioning and allocation have to happen simultaneously, since the values (i.e.,  $v_{ij}$ ) and whether it contributes positively or negatively (i.e.,  $v_{ij} - \rho_j$ ) depend on the buyer and bundle under consideration. We show how algorithms for GAP (generalized assignment problem) with matroid constraints [13] can be used.

**Relax-and-Round.** In order to extend our results from the offline to the online settings, and to add in side-constraints, we then consider linear programming (LP) based relax-and-round algorithms for the AVA problem. The LP relaxations take advantage of the structural properties above, as they try to capture the best bundling-based algorithms (and hence to approximate the optimal solution of any kind). Once we have fractional solutions to the LP, we can then round these in both offline and online settings to get our feasible allocations.

Our first rounding-based algorithm, given in §4, is in the offline setting, and yields another  $O(1)$ -approximate algorithm for AVA, qualitatively matching the result from Theorem 2. While the constants are weaker, the result illustrates our ideas, and allows us to support additional side-constraints (more on this in §1.1.1).

**Online Algorithms.** We then turn to online AVA, where items arrive over  $T$  timesteps, and must be allocated to buyers as soon as they arrive. We want to maintain feasible solutions to the AVA at all times. We show that under adversarial arrivals, only trivial  $O(T)$  approximations are possible. This forces us to focus our attention on i.i.d. arrivals. Our first result is a time-efficient approximation of the optimum (computationally-unbounded) online algorithm:

► **Theorem 3** (Online AVA: Approximating the Optimal Online IID Algorithm). *There exists a randomized polynomial-time online algorithm for the AVA problem which achieves a constant factor of the value achieved by the optimal (computationally-unbounded) online algorithm.*

To approximate the optimum online algorithm, we provide an LP capturing a constraint only applicable to online algorithms, inspired by such constraints from the secretary problem and prophet inequality literatures [12, 34]. We then provide a two-phase online algorithm achieving a constant approximation of this LP, analyzed via a coupling with an imaginary algorithm that may violate AVA constraints and allocate items to several buyers.

We then turn our attention to approximating the ex-post optimum (a.k.a., getting a competitive ratio for the observed sequence). In contrast, we show that when comparing with the ex-post optimum, no such constant approximation ratio is possible, but we give matching upper and lower bounds. (Due to lack of space, this is deferred to Appendix A.)

► **Theorem 4** (Online AVA: Ex-post Guarantees (Informal)). *There exist families of online i.i.d. AVA instances on which any online algorithm is  $\Omega\left(\frac{\log T}{\log \log T}\right)$ -competitive. In contrast, there exists an online algorithm matching this bound asymptotically (on all instances).*

The lower bound is proved by giving an example using a balls-and-bins process (and its anti-concentration). Then we formulate an LP capturing this kind of anti-concentration, using which we match the lower bound, under some mild technical conditions (see Appendix A for details).

### 1.1.1 Generalizations

There are many interesting generalizations of the basic problem. For example, there might exist “budgets” which limit the number of items any buyer can receive; or more generally we may have costs on items which must sum to at most the buyer’s budgets. These costs could be different for different buyers, and in different units than those captured by constraint (1.1). These constraints are the natural ones considered in packing problems; in general, we can consider the AVA constraint as being a non-packing constraint on the allocation that can be supplemented with other conventional packing constraints. As we show in §4.3, our relax-and-round algorithm extends seamlessly to accommodate such side constraints, provided any individual item has small cost compared to the relevant budgets.

Another natural generalization is *return-on-spend* (RoS) constraints, which have been central to much recent work on advertisement allocation (see [25, 20]) and §1.2). We call the problem *generalized AVA* (GenAVA) and define it as follows: the objective is to maximize social welfare, but now the average value is measured in a more general way. Indeed, the allocation of item  $i$  to buyer  $j$  can incur a different “cost”  $c_{ij}$ , and the average-value constraint becomes the following ROS constraint:

$$\forall j, \quad \sum_i v_{ij} x_{ij} \geq \rho_j \cdot \left( \sum_i c_{ij} x_{ij} \right). \quad (1.2)$$

In contrast to AVA, we show that allowing general costs  $c_{ij}$  in the generalized AVA problem in (1.2) makes it as hard as one of the hardest combinatorial problems – computing a maximum clique in a graph. In particular, we show that it is NP-hard to  $n^{1-\varepsilon}$ -approximate GenAVA with  $n$  buyers, for any constant  $\varepsilon > 0$ . In Appendix B we show that similar hardness persists even for stochastically generated inputs, and the problem remains hard even if we allow for bicriteria approximation.

## 1.2 Related Work

Resource allocation is one of the most widely-studied topics in theoretical computer science. Here we briefly discuss some relevant lines of work.

**Packing/Covering Allocation Problems.** The *budgeted allocation problem* or ADWORDS of [32] is NP-hard to approximate within some constant [14], and constant approximations are known even online [32, 11, 28]. The *generalized assignment problem (GAP)* [22] and its extension, the *separable assignment problem*, have constant approximations in both offline [23, 13] and (stochastic) online settings [30]. In both cases, arbitrarily-good approximations are impossible under adversarial online arrivals, even under structural assumptions allowing for an offline PTAS (e.g., “small” bids) [32]. However, assuming both small bids and *random-order* (or *i.i.d.*) arrivals allows us to achieve  $(1 - \varepsilon)$ -competitiveness [16, 18, 30, 26, 2]. Some such allocation problems are also considered with concave or convex utilities [17, 7]. As noted above, many results and techniques for (offline and online) packing and covering constraints are not applicable to our problem, which is neither a packing nor covering problem in the conventional sense.

**RoS constraints in online advertising.** Return-on-spend constraints as defined in (1.2) have received much attention in recent years in the context of online advertising. Several popular autobidding products allow advertisers to provide campaign-level RoS constraints with a goal to maximize their volume or value of conversions (sales) [25, 20]). Fittingly, there has been

much interest in understanding the RoS setting along various directions, including optimal bidding [1], mechanism design [8, 24], and on welfare properties at equilibrium [1, 15, 31]. In these results, distributed bidding based algorithms are shown to achieve a constant fraction of the optimal welfare. However, note that the per-item costs in the autobidding setting are *endogenous* (set via auction dynamics) whereas in our allocation problem there is no pricing mechanism and the costs are *exogenous*. Our results about the hardness of the generalized AVA show that under exogenous prices, such allocation problems do not admit constant (or even sublinear) approximation guarantees.

**Approximating the optimum online algorithm.** Our online i.i.d. results relate to a recent burgeoning line of work on approximation of the optimum online algorithm via restricted online algorithms. This includes restriction to polynomial-time algorithms (as in our case) [34, 33, 10, 3, 29], fair algorithms [5], order-unaware algorithms [19] and inflexible algorithms [4, 35], and more. These works drive home the message that approximating the optimum online algorithm using restricted algorithms is hard, but can often lead to better approximation than possible when comparing to the (unattainable) benchmark of the ex-post optimum. We echo this message, showing that for our problem under i.i.d. arrivals, a constant-approximation of the optimum online algorithm (using polytime algorithms) is possible, but is impossible when comparing to the optimum offline solution.

### 1.3 Problem Formulation

In the *average-value-constrained allocation problem (AVA)*, allocating item  $i$  to buyer  $j$  yields a value of  $v_{ij}$ . Each buyer  $j$  requires that the average value they obtain from allocated items be at least  $\rho_j$ . We wish to (approximately) maximize the total social welfare, or sum of values obtained by the buyers, captured by the following integer LP:

$$\begin{aligned}
 \max \quad & \sum_{(i,j) \in E} v_{ij} x_{ij} && \text{(AVA-ILP)} \\
 \text{s.t.} \quad & \sum_i v_{ij} x_{ij} \geq \rho_j \cdot \sum_i x_{ij} && \forall \text{ buyers } j \\
 & \sum_j x_{ij} \leq 1 && \forall \text{ items } i \\
 & x_{ij} \in \{0, 1\} && \forall \text{ items } i, \text{ buyers } j.
 \end{aligned}$$

An instance  $\mathcal{I}$  of AVA can be captured by a bipartite graph  $(I, J, E)$ , with a set  $I$  of items and set  $J$  of buyers, and edges  $E \subseteq I \times J$ , capturing all buyer-item pairs with non-zero value. For  $i \in I$  and  $j \in J$ , edge  $(i, j)$  has value  $v_{ij}$ . We say edge  $(i, j)$  is a *P-edge* (positive edge) if it has non-negative *excess*  $v_{ij} - \rho_j \geq 0$ , and an *N-edge* otherwise, in which case we refer to  $v_{ij} - \rho_j < 0$  as its *deficit*. An item  $i$  is a *P-item* if *all* its edges in  $E$  are *P-edges*, and an *N-item* if *all* its edges in  $E$  are *N-edges*: naturally, some items may be neither *P-items* or *N-items*. We will call an instance *unit- $\rho$*  if  $\rho_j = 1$  for all buyers.<sup>2</sup>

In the online setting, the  $n$  buyers and their  $\rho_j$  values are known a priori, but items  $i$  are revealed one at a time, together with their value  $v_{ij}$  for each buyer  $j$ , and an algorithm must decide what buyer to allocate an item to (if any), immediately and irrevocably on

<sup>2</sup> Such instances capture the core difficulty of the AVA problem, and our examples (except those for GenAVA in Section B) are unit- $\rho$  instances, so one can WLOG take  $\rho_j = 1$  in the first read.

arrival. In the online i.i.d. setting,  $T$  items are drawn (one after another) i.i.d. from a known distribution over  $m$  known item types, with type  $i$  drawn with probability  $q_i$ . We say an edge type  $(i, j)$  is an  $N$ -edge type or a  $P$ -edge type if  $v_{ij} - \rho_j < 0$  or  $v_{ij} - \rho_j \geq 0$ , respectively.

## 1.4 Paper Outline

We begin in §2 by proving some structural lemmas regarding AVA, including an unintuitive non-linear dependence of the welfare on the amount of supply. In §3 we present the improved algorithm for the offline setting giving Theorem 2. In §4 we present our LP-rounding algorithm for AVA in an offline setting. We also discuss the approach's extendability, allowing to incorporate additional constraints, in §4.3. Building on this offline rounding-based algorithm, in §5 we present a constant-approximation of the optimum online algorithm. In the interest of space, we defer the discussion of competitive ratio bounds to Appendix A, and our hardness results to Appendix B.

## 2 The Structure of Near-optimal Solutions for AVA

In this section, we show how to partition any feasible allocation of AVA instances into structured subsets (which we call *permissible bundles*). This bundling-based structure will prove useful for all of our algorithms.

► **Definition 5** (Bundling). *A set  $S$  of edges incident on buyer  $j$  is a permissible bundle if **nolistsep**  $S$  consists of a single  $P$ -edge  $(i^*, j)$  and zero or more  $N$ -edges  $(i, j)$ , and **nolistsep** the edges in  $S$  satisfy the average-value constraint, i.e.,  $\sum_{(i,j) \in S} v_{ij} \geq \rho_j \cdot |S|$ . A bundling-based solution is one that can be partitioned into a collection of permissible bundles.*

Clearly, no bundling-based solution can be better than the best unconstrained solution, but in the following lemma we show a converse, up to constant factors. (Throughout, we use the shorthand notation  $v \cdot x := \sum_{ij} v_{ij} x_{ij}$  for any vector  $x \in \mathbb{R}^E$ .)

► **Lemma 6** (Good Bundling-Based Solution). *Let  $x^*$  be a solution to an instance of AVA. Then, there exists a bundling-based solution  $\hat{x}$  of value at least  $v \cdot \hat{x} \geq \frac{1}{2} v \cdot x^*$ .*

As a corollary, the best bundling-based solution is a 2-approximation, and so we will strive to approximate such bundling-based solutions.

We prove a strengthening of Lemma 6 which also addresses online settings.

► **Definition 7** (Committed Bundling). *An online algorithm is a committed bundling-based algorithm if its solution consists of permissible bundles, and items can only be added to bundles; in particular, it commits to the allocation of each item to a particular bundle, and does not move items between permissible bundles.*

► **Lemma 8** (Online Bundling-Based Solution). *Let  $x^*$  be a solution to an instance of AVA, with  $x^*$  revealed online and (all interim partial solutions) satisfying the average-value constraints throughout. Then there exists a solution  $\hat{x}$  that is the output of a committed online bundling-based algorithm, of value at least  $v \cdot \hat{x} \geq \frac{1}{2} v \cdot x^*$ .*

**Proof.** For each buyer  $j$ , consider the edges  $S := \{(i, j) \mid x_{ij}^* = 1\}$  corresponding to items assigned to buyer  $j$  in solution  $x^*$ , in order of addition to the solution  $x^*$ , namely  $e_1, e_2, \dots, e_{|S|}$ , with  $e_k = (i_k, j)$ . We now show how a committed online algorithm can output a collection of permissible bundles of at least half the value from among the edges in  $S$ ; doing this for each buyer proves the result.

Consider  $i_k$ , i.e., the  $k$ -th item allocated to  $j$  by  $x^*$ , if  $e_k$  is a  $P$ -edge (i.e.  $v_{i_k,j} \geq \rho_j$ ), we denote  $p = i_k$ , open (create) a bundle  $B_p = \{(j,p)\}$  and allocate appropriately in the new solution  $\hat{x}$ . When  $e_k = (i_k,j)$  is an  $N$ -edge, if  $e_k$  can be added to some open bundle  $B_p$  of  $j$  while keeping it permissible, we add  $(i_k,j)$  to  $B_p$  in solution  $\hat{x}$ ; otherwise, we pick some open bundle  $B_p$  of  $j$  and mark it as closed (and never add more edges to this bundle). Since  $x^*$  is feasible throughout the online arrival, for any  $k \in [1, |S|]$  we have that  $\sum_{\ell \leq k} v_{i_\ell,j} \geq k \cdot \rho_j$ , and since we allocate all  $P$ -edges of  $x^*$  in  $\hat{x}$  and only allocate a subset of the  $N$ -edges, we find that there must always be some open bundle of  $j$  when considering an  $N$ -edge  $e_k$ . Therefore, the above (committed) bundling-based online algorithm is well-defined. Now, each bundle is closed by at most one  $N$ -edge  $(i,j)$ , and so we can charge the  $N$ -edges  $(i,j)$  allocated in  $x^*$  but not in  $\hat{x}$  to the  $P$ -edge  $(p,j)$  in the bundle  $B_p$  that they closed. But by definition of the  $P$ -edge and  $N$ -edge, we know  $v_{pj} \geq \rho_j \geq v_{ij}$ . Therefore, denoting by  $x_D^*$  the part of the solution  $x^*$  that is *discarded* in  $\hat{x}$  and by  $x_p^*$  and  $x_n^*$  the value of the  $P$ -edges and  $N$ -edges allocated by both  $x^*$  and the new solution  $\hat{x}$ , we have that  $v \cdot x_D^* \leq v \cdot x_p^*$ . Hence,

$$v \cdot x^* = v \cdot x_D^* + v \cdot (x^* - x_D^*) \leq 2v \cdot x_p^* + x_n^* \leq 2v \cdot (x_p^* + x_n^*). \quad (2.3)$$

That is, the obtained bundles of the solution  $\hat{x} = x_p^* + x_n^*$  constitute a 2-approximation. ◀

► **Remark 9.** This loss of a factor of two in the value is tight. To see this, consider a single-buyer unit- $\rho$  AVA instance. There are  $\frac{1}{\varepsilon}$   $N$ -edges each with value  $1 - \varepsilon$  and  $\frac{1}{\varepsilon(1-\varepsilon)}$   $P$ -edges each with value  $1 + \varepsilon(1 - \varepsilon)$ . It is feasible to allocate all items to the buyer, and (arbitrarily close to) half the value of this solution is given by  $N$ -edges, but any permissible bundle contains no  $N$ -edges as any single  $P$ -edge doesn't have enough excess to cover the deficit of any  $N$ -edge.

For our algorithms it will be convenient if each item is incident only on  $P$ -edges, or only on  $N$ -edges, thus removing the ambiguity about whether to use these as the single  $P$ -edge in a permissible bundle. Fittingly, we call such instances *unambiguous*. For example, when all buyers have the same average-value constraint (i.e.  $\forall j : \rho_j = \rho$ ), for any item  $i$  incident on a  $P$ -edge (i.e.,  $\exists j : v_{ij} \geq \rho$ ), we can trivially drop all  $N$ -edges of the item (i.e., drop  $(i,j')$  where  $v_{ij'} < \rho$ ) since there is no reason to allocate any  $N$ -edge instead of a  $P$ -edge of  $i$ , and so making such instances unambiguous comes with no cost. As we now show, any instance of AVA in general can be made unambiguous while still preserving a bundling-based allocation that is constant-approximate for the original instance.

► **Lemma 10 (Bundling Unambiguous Sub-Instances).** *Given an AVA instance  $\mathcal{I} = (I, J, E)$ , dropping all of the  $P$ -edges or all the  $N$ -edges of each item  $i \in I$  independently with probability  $1/2$  results in an unambiguous sub-instance  $\mathcal{I}' = (I, J, E')$  (where  $E' \subseteq E$ ), admitting a bundling-based solution  $x'$  which is 4-approximate for  $\mathcal{I}$ .*

**Proof.** Let  $x^*$  be an optimal solution for  $\mathcal{I}$ . If we denote by  $x_p^*$  and  $x_n^*$  the characteristic vector for  $P$ -edges and  $N$ -edges allocated by both  $x^*$  and  $\hat{x} = x_p^* + x_n^*$  as in the proof of Lemma 8, then, by the penultimate inequality of Equation (2.3), we have that  $v \cdot x^* \leq 2v \cdot x_p^* + v \cdot x_n^*$ . Now, consider the solution  $x'$  consisting of all  $P$ -edges allocated in  $\hat{x}$  that were not dropped and all non-dropped  $N$ -edges allocated in bundle  $S$  whose  $P$ -edge was also not dropped. We therefore have that this new solution has value precisely  $\frac{1}{2}v \cdot x_p^* + \frac{1}{4}v \cdot x_n^*$ , and so, by Equation (2.3), we have that  $x'$  is a 4-approximation, since

$$v \cdot x^* \leq 4 \cdot \left( \frac{1}{2}v \cdot x_p^* + \frac{1}{4}v \cdot x_n^* \right) = 4v \cdot x'. \quad \blacktriangleleft$$

We also provide an alternative, deterministic method to find such an unambiguous subinstance. However, since our algorithms are randomized, we defer discussion of this method to the full version. Note in unambiguous instances, every item is either a  $P$ -item or an  $N$ -item.

## 2.1 Welfare is non-linear in supply

In this section we provide a bound on the multiplicative gain in welfare in terms of increased supply. This will prove useful later. For now, it illustrates non-linearity of the AVA problem in its supply. (This is in contrast to other allocation problems where the welfare is at best linear in the supply.)

To motivate this bound, consider the outcome of creating  $k$  copies of each item in an AVA instance. Clearly, the welfare increases by a factor of at least  $k$ , as we can just repeat the optimal allocation for the original instance  $k$  times. However, as the following example illustrates, welfare can be *super-linear* in the supply size increase for AVA.

► **Example 11.** Consider a unit- $\rho$  instance of  $k$ -buyer AVA with a single  $P$ -item of value  $1 + k\varepsilon$  for all buyers and  $k$  many  $N$ -items, with the  $i$ -th  $N$ -items having value zero for all buyers except for one distinct buyer  $i$ , to whom it has value  $1 - \varepsilon$ . In this instance  $\text{OPT} \approx 2$ , since the  $P$ -item can only be allocated to a single buyer, who can then only be allocated one  $N$ -item, while in the instance obtained by creating  $k$  copies of each item we can allocate a  $P$ -item to each buyer together with  $k$  many  $N$ -items, and so for this instance  $\text{OPT} \approx k^2$ , i.e., increasing supply  $k$ -fold increases the welfare  $(k^2/2)$ -fold.

The following lemma shows that the above example is an extreme case, and for a  $k$ -fold increase in supply, an  $O(k^2)$ -fold increase in welfare is best possible.

► **Lemma 12 (Supply Lemma).** *Let  $\mathcal{I} = (I, J, E)$  be an AVA instance, and let  $\mathcal{I}' = (I', J, E')$  be an instance with the same buyer set and underlying costs and values obtained by copying each item in  $\mathcal{I}$  some  $k$  times.*

$$\text{OPT}(\mathcal{I}') \leq O(k^2) \cdot \text{OPT}(\mathcal{I}).$$

**Proof.** Since bundling-based solutions are nearly optimal up to a constant factor of 2, we can start with an optimal bundling-based allocation  $\mathcal{A}'$  for  $\mathcal{I}'$  and randomly (and independently) associate the items of  $\mathcal{I}$  with one of their  $k$  copies in  $\mathcal{I}'$ , allocating them as in  $\mathcal{A}'$ . Finally, we remove all non-permissible obtained bundles to obtain allocation  $\mathcal{A}$  for  $\mathcal{I}$ . For each copy  $i'$  of an item  $i$ , if  $i'$  is allocated in a  $P$ -edge in  $\mathcal{A}'$ , the probability that  $i$  is associated with  $i'$  (and thus assigned to the same buyer by  $\mathcal{A}$ ) is precisely  $1/k$ . In contrast, if  $i'$  is allocated in an  $N$ -edge by  $\mathcal{A}'$ , the probability that  $\mathcal{A}$  allocates  $i$  the same way as  $i'$  is precisely  $1/k^2$ , as this requires both  $i$  to be assigned to the same bundle (associated with the same copy) and the  $P$ -edge of this bundle to similarly be assigned to the same bundle. The lemma then follows by linearity of expectation. ◀

## 3 Offline Algorithm via Reduction to Matroid-Constrained GAP

In this section we provide an improved constant-approximation for AVA in the *offline* setting; we will show in Appendix B.1 that the problem is hard to approximate to better than  $\frac{e}{e-1}$ .

► **Theorem 13.** *There exists a  $(\frac{4e}{e-1} + o(1))$ -approximate randomized algorithm for AVA.*

The algorithm proceeds by reducing AVA to GAP with matroid constraints. Recall that an instance of the *generalized assignment problem* (GAP) consists of  $n$  elements that can be packed into  $m$  bins. Packing an element  $e$  into a bin  $b$  gives a value  $v_{eb}$  and uses up  $s_{eb}$  space in that bin. If we let  $y_{eb} \in \{0, 1\}$  denote the indicator for whether element  $e$  is assigned to

bin  $b$ , then naturally  $\sum_b y_{eb} \leq 1$ . Each bin has unit size, and so the size of elements assigned to bin  $b$  is at most 1: in other words,  $\sum_e s_{eb} y_{eb} \leq 1$ . The goal is to maximize the total value of the assignment  $\sum_{eb} v_{eb} y_{eb}$ . [23] gave a  $(1 - 1/e)$ -approximation for this problem. [13] gave the same approximation for an extension of the problem, where the opened subset of bins must be an independent set in some given matroid  $\mathcal{M}$ .

► **Theorem 14.** *There exists a randomized polynomial-time algorithm that, for any unambiguous AVA instance, outputs a solution with expected value at least  $(1 - 1/e - o(1))$  times the optimal bundling-based solution.*

**Proof.** Given an unambiguous AVA instance (i.e., one where each item is incident on only  $P$ -edges or only  $N$ -edges), we construct an instance of Matroid-Bin GAP as follows:

1. *Elements and bins:* For each  $P$ -item  $p$  and buyer  $j$ , construct a bin  $(p, j)$  in the GAP instance. The elements of the GAP instance are exactly the items of the AVA instance.
2. *Values/sizes of  $P$ -items:* Assigning a  $P$ -item  $p$  to bin  $(p, j)$  yields value  $v_{pj}$  and uses zero space; Assigning  $P$ -item  $p$  to a bin  $(p', j)$  with  $p \neq p'$  yields value zero and uses  $1 + \varepsilon$  space.
3. *Values/sizes of  $N$ -items:* Assigning  $N$ -item  $i$  to bin  $(p, j)$  yields value  $v_{ij}$  and uses  $\frac{\rho_j - v_{ij}}{v_{pj} - \rho_j}$  space.
4. *Matroid on the bins:* Finally, the matroid  $\mathcal{M}$  on the bins is a partition matroid, requiring that we choose at most one bin from  $\{(p, j) \mid j \in B\}$ , for each  $P$ -item  $p$ .

The construction above results in a value-preserving one-to-one correspondence between feasible GAP solutions which are *maximal*, i.e., where each  $P$ -item  $p$  is assigned to some bin, and permissible bundling-based solutions to the AVA instance. Indeed, for any feasible bundling-based solution to the AVA instance, fix a bundle  $(p, j)$  containing the item set  $S$ . The value of placing the items in  $S$  in the bin  $(p, j)$  is precisely  $\sum_{i \in S} v_{ij}$ . Summing over all bins, we find that both solutions (to the AVA and GAP instance) have the same value. On the other hand, the GAP solution is feasible since for each  $P$ -item  $p$  we open up at most one bin  $(p, j)$  (thus respecting the matroid constraint) and moreover each bin's size constraint is respected due to the per-bundle average-value constraint and the zero size of  $p$  in bin  $(p, j)$ , implying that  $\sum_{i \in S} s_{i,(p,j)} = \sum_{i \in S \setminus \{p\}} \frac{\rho_j - v_{ij}}{v_{pj} - \rho_j} \leq 1$ . Similarly, starting with a maximal solution to the GAP instance, the single bin  $(p, j)$  into which  $p$  is placed has its average-value constraint satisfied (note that  $p$  cannot be placed in a bin  $(p', j)$  for  $p' \neq p$ , where its size is  $1 + \varepsilon$ ), and the value of the bundles obtained this way is the same as the GAP solution's value. Now the  $(1 - 1/e - o(1))$ -approximation algorithm for GAP with matroid constraints [13] gives the same approximation for AVA on unambiguous instances. ◀

Theorem 14 combined with Lemma 10 completes the proof of Theorem 13.

## 4 An Offline Algorithm via Relax-and-Round

Let us now present an LP-rounding based algorithm for AVA. This more sophisticated algorithm yields another constant-approximate offline algorithm, which also allows to incorporate additional side constraints, see Section 4.3). Moreover, this section's algorithm also provides a template for our main *online* algorithms.

The natural starting point for an LP-rounding based algorithm, the LP relaxation obtained by dropping the integrality constraints of (AVA-ILP), turns out to be a dead end. This relaxation has an integrality gap of  $\Omega(n)$  on  $n$ -buyer instances,<sup>3</sup> even for unit- $\rho$ , as shown by the reinspecting the instance of Example 11.

<sup>3</sup> Recall that an LP relaxation's *integrality gap* is the difference in objective between its best fractional and integral solutions.

## 13:10 The Average-Value Allocation Problem

► **Example 15.** Consider an  $n$ -buyer unit- $\rho$  instance with a single  $P$ -item  $p$  of value  $1 + n\varepsilon$  for all buyers, and  $n$   $N$ -items, with the  $i$ -th  $N$ -item having zero value for all buyers except for buyer  $j_i$ , for whom its value is  $1 - \varepsilon$ . An assignment  $x_{pj} = \frac{1}{n}$  for all buyers  $j$  and  $x_{ij_i} = 1$  for every  $N$ -item  $i$  gives value  $n + 1$  for the LP relaxation of (AVA-ILP), while clearly the optimal integral solution has value  $\approx 2$ .

Therefore, to obtain any constant approximation via LP rounding, we need a tighter relaxation. To this end, we rely on Lemmas 6 and 10, and provide the following relaxation for *bundling-based* solutions for unambiguous AVA instances. This LP has decision variables  $x_{ijp}$  for ( $P$  or  $N$ )-item  $i$ , buyer  $j$  and  $P$ -item  $p$ . Informally, these correspond to the probability that  $i$  is allocated to  $j$  in the bundle with  $P$ -item  $p$ , which we denote by  $jp$ . (Note: this polynomially-sized LP is clearly poly-time solvable.)

$$\max \sum_{i,j,p} v_{ij} x_{ijp} \quad (\text{Bundle-LP})$$

$$\text{s.t.} \quad \sum_i (\rho_j - v_{ij}) x_{ijp} \leq 0 \quad \forall j, p \quad (4.4)$$

$$\sum_{j,p} x_{ijp} \leq 1 \quad \forall i \quad (4.5)$$

$$x_{ijp} \leq x_{pjp} \quad \forall i, j, p \quad (4.6)$$

$$x_{p'jp} = 0 \quad \forall j, P\text{-item } p' \neq p \quad (4.7)$$

$$x_{ijp} \geq 0 \quad \forall i, j, p$$

Intuitively, the bundling, and in particular Equation (4.6), will allow us to overcome the integrality gap example above. We formalize this intuition later by approximately rounding this LP, but first we show that (Bundle-LP) is a relaxation of bundling-based allocations for unambiguous AVA instances.

► **Lemma 16.** *For any unambiguous AVA instance, the value of (Bundle-LP) is at least as high as that of the optimal bundling-based allocation.*

**Proof.** Fix a (randomized) bundling-based allocation algorithm  $\mathcal{A}$ . Let  $Y_{ijp}$  be the indicator for  $\mathcal{A}$  having allocated item  $i$  in bundle  $jp$ . We argue that  $Y_{ijp}$  satisfy the constraints of (Bundle-LP), realization by realization. Consequently, by linearity of expectation, so do their marginals,  $\mathbb{E}[Y_{ijp}]$ . Constraint (4.4) holds since  $\mathcal{A}$  satisfies the average-value constraint for each bundle. Constraint (4.5) holds since each item is allocated at most once. Constraint (4.6) holds because bundle  $jp$  must be opened for  $i$  to be allocated in it. Constraint (4.7) holds since permissible bundles have a single  $P$ -item in them. Finally, non-negativity of  $\mathbf{Y}$  is trivial. We conclude that  $\mathbb{E}[\mathbf{Y}]$  is a feasible solution to the above LP, with objective precisely  $\sum_{i,j,p} v_{ij} \mathbb{E}[Y_{ijp}]$ . The lemma follows. ◀

We now turn to rounding this LP. To this end, we consider a two-phase algorithm, whose pseudo-code is given in Algorithm 1. In Phase I we *open bundles*, letting each  $P$ -item  $p$  pick a single buyer  $j$  with probability  $x_{pjp}$ ,<sup>4</sup> and opening the bundle  $jp$ . In Phase II we *enrich the bundles*, by adding  $N$ -items to them. Specifically, for each  $N$ -item  $i$ , we create a set  $S_i$  containing each open bundle  $jp$  independently with probability  $\alpha \cdot \frac{x_{ijp}}{x_{pjp}}$ , where  $\alpha \in [0, 1]$

<sup>4</sup> Since Constraint (4.5) is tight for every  $P$ -item in any optimal LP solution,  $\{x_{pjp}\}_j$  is a distribution over buyers.



is a parameter to be specified later. Then, if this set  $S_i$  contains a single bundle  $jp$  and adding  $i$  to this bundle would not violate the average-value constraint restricted to the bundle (denoted by  $\text{BundleAV}_{jp}$ ), i.e., this bundle would remain permissible, then we allocate  $i$  to the bundle  $jp$ . Otherwise, we leave  $i$  unallocated.

■ **Algorithm 1** Offline rounding of Bundle-LP.

---

```

1: Make the instance unambiguous as in Lemma 10
2: Let  $\mathbf{x}$  be an optimal solution to (Bundle-LP) for the obtained unambiguous instance
3: for each  $P$ -item  $p$  do ▷ Phase I
4:   Pick  $j$  according to distribution  $\{x_{pjp}\}_{j=1,\dots,n}$  and open bundle  $jp$ 
5: for each  $N$ -item  $i$  do ▷ Phase II
6:    $S_i \leftarrow \emptyset$ 
7:   for each bundle  $jp$ , with probability  $\alpha \cdot \frac{x_{ijp}}{x_{pjp}}$  do
8:     if  $jp$  was opened in Phase I then
9:        $S_i \leftarrow S_i \cup \{jp\}$ 
10:  if  $|S_i| = 1$  then
11:    if the only bundle  $jp \in S_i$  remains permissible after adding  $i$  to it then
12:      Allocate  $i$  to  $jp$ 

```

---

Algorithm 1 clearly outputs a feasible allocation, since it only allocates  $N$ -items  $i$  to a bundle  $jp$  if this would not violate the average-value constraint of the bundle, and hence by linearity the average-value constraint of the buyer remains satisfied. Moreover, the algorithm is well-defined; in particular, the probability spaces defined in lines 4 and 7 are valid, by constraints (4.5) for  $P$ -item  $p$ , and (4.6) for triple  $i, j, p$ , respectively. We turn to analyzing this algorithm's approximation ratio. For this, we will lower bound the probability of each item  $i$  to be allocated in bundle  $jp$  in terms of  $x_{ijp}$ .

By Section 4, each  $P$ -item  $p$  is assigned in bundle  $jp$  precisely with probability  $x_{pjp}$ . Consequently, the expected value Algorithm 1 obtains from  $P$ -items is precisely their contribution to the LP solution's value. It remains to understand what value we get from  $N$ -items.

#### 4.1 Allocation of $N$ -items

To bound the contribution of  $N$ -items, we consider any tuple of  $N$ -item  $i$ , buyer  $j$  and  $P$ -item  $p$ . Note that  $N$ -item  $i$  is assigned to bundle  $jp$  if and only if all the four following events occur:

1.  $\mathcal{E}_1$ : the event that bundle  $jp$  is open, which happens with probability  $x_{pjp}$ .
2.  $\mathcal{E}_2$ : the event that the Bernoulli( $\alpha \cdot \frac{x_{ijp}}{x_{pjp}}$ ) in Section 4 comes up heads for  $jp$ .
3.  $\mathcal{E}_3$ : the event that  $S_i \setminus \bigcup_{j'=1,\dots,n} \{j'p\} = \emptyset$ .
4.  $\mathcal{E}_4$ : the event that  $jp$  would remain permissible if we were to add  $i$  to bundle  $jp$ .

We note that events  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$  are all independent, as they depend on distinct (and independent) coin tosses. So, for example,  $\Pr[S_i \ni jp] = \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2] = \alpha \cdot x_{ijp}$ . Moreover, we have the following simple bound on  $\Pr[\mathcal{E}_3]$ .

► **Lemma 17.**  $\Pr\left[\bigwedge_{\ell=1}^3 \mathcal{E}_\ell\right] = \prod_{\ell=1}^3 \Pr[\mathcal{E}_\ell] \geq (1 - \alpha) \cdot \alpha \cdot x_{ijp}$ .

## 13:12 The Average-Value Allocation Problem

**Proof.** The first equality follows from independence of  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ . We therefore turn to lower bounding  $\Pr[\mathcal{E}_3]$ . Since  $\Pr[X > 0] \leq \mathbb{E}[X]$  for any integer random variable  $X \geq 0$ , we know

$$\Pr[\overline{\mathcal{E}_3}] \leq \mathbb{E} \left[ \left| S_i \setminus \bigcup_{j'} \{j'p\} \right| \right] = \sum_{p' \neq p} \sum_{j'} \alpha \cdot x_{ij'p'} \leq \alpha,$$

where the equality follows from  $\Pr[S_i \ni j'p'] = \alpha \cdot x_{ij'p'}$  by the above, and the last inequality follows from Constraint (4.5). Since  $\Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2] = \alpha \cdot x_{ijp}$ , the lemma follows.  $\blacktriangleleft$

**A challenge.** As noted above,  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$  are independent, resulting in a simple analysis for the probability  $\Pr[\bigwedge_{\ell=1}^3 \mathcal{E}_\ell] = \prod_{\ell=1}^3 \Pr[\mathcal{E}_\ell]$ . Unfortunately, lower bounding  $\Pr[\mathcal{E}_4 \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3]$  is more challenging, due to possible *negative correlations* between  $\mathcal{E}_4$  and  $\mathcal{E}_3$ . To see this, note that  $\mathcal{E}_3 \wedge \mathcal{E}_1$  implies  $S_i = \{jp\}$ , and this event can be positively correlated with previous  $N$ -items  $i'$  having  $S_{i'} = \{jp\}$ , thus making it more likely that  $jp$  won't be able to accommodate  $i$  under  $\text{BundleAV}_{jp}$ .

We can overcome this challenge of negative correlations, provided  $(i, j)$  has small deficit compared to  $(p, j)$ 's excess. (We address the large deficit case separately later.) Specifically, by coupling our algorithm with an algorithm that allocates more often and does not suffer from such correlations, we can lower bound this conditional probability as follows.

**► Lemma 18.** *Let  $\beta \in [0, 1]$ . If  $i, j, p$  are such that  $\rho_j - v_{ij} \leq \beta \cdot (v_{pj} - \rho_j)$ , then*

$$\Pr[\mathcal{E}_4 \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq 1 - \frac{\alpha}{1 - \beta}.$$

**Proof.** Consider an imaginary algorithm  $\mathcal{A}'$  that allocates every  $N$ -item  $i'$  into every bundle  $j'p' \in S_{i'}$ , even when  $|S_{i'}| > 1$  (so we may over-allocate) and even if this violates the  $\text{BundleAV}_{j'p'}$  constraint. Coupling  $\mathcal{A}'$  with Algorithm 1 by using the same randomness for both algorithms, we have that item  $i'$  is allocated to bin  $j'p'$  by  $\mathcal{A}'$  with probability precisely  $\Pr[S_{i'} \ni j'p'] = \alpha \cdot x_{i'j'p'}$ . In particular,  $\mathcal{A}'$  only allocates more items than Algorithm 1.

We denote by  $N'_{jp}$  the set of  $N$ -items allocated to bundle  $jp$  by  $\mathcal{A}'$ . Now, let  $\mathcal{E}'_4$  be the event that  $\sum_{i' \in N'_{jp} \setminus \{i\}} (\rho_j - v_{i'j}) \leq (1 - \beta) \cdot (v_{pj} - \rho_j)$ , that is, the deficit of  $N$ -items other than  $i$  that  $\mathcal{A}'$  allocated to the bundle  $jp$  together only consumes at most a  $(1 - \beta)$  fraction of  $p$ 's excess for  $j$ . By the small deficit assumption on  $i, j, p$ , we know that event  $\mathcal{E}'_4$  is sufficient for  $\text{BundleAV}_{jp}$  to be satisfied if Algorithm 1 were to add  $i$  to  $jp$ . Thus,  $\mathcal{E}'_4$  implies  $\mathcal{E}_4$  in any realization (of the randomness), since  $\mathcal{A}'$  only allocates more items to each bin than Algorithm 1. On the other hand, we also have that both  $\mathcal{E}'_4$  and  $\mathcal{E}_1$  are independent of both  $\mathcal{E}_2 \wedge \mathcal{E}_3$ , since the latter combined event depends on an independent random coin toss ( $\mathcal{E}_2$ ) and events concerning other bundles  $j'p'$ , which are both independent of the randomness concerning bundle  $jp$ . (Here we use that  $\mathcal{A}'$  allocates  $i$  to  $jp$  whenever  $S_i \ni jp$ , regardless of other bundles  $j'p'$  belonging to  $S_i$ .) Consequently, by standard applications of Bayes' Law, we obtain the following.

$$\Pr[\mathcal{E}'_4 \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = \Pr[\mathcal{E}'_4 \mid \mathcal{E}_1].$$

As the imaginary algorithm  $\mathcal{A}'$  assigns  $i'$  to  $jp$  (i.e.  $i' \in N'_{jp}$ ) iff  $S_{i'} \ni jp$ , we know that

$$\begin{aligned} \mathbb{E} \left[ \sum_{i' \in N'_{jp}} (\rho_j - v_{i'j}) \mid \mathcal{E}_1 \right] &= \sum_{i' \neq p} (\rho_j - v_{i'j}) \cdot \Pr[S_{i'} \ni jp \mid \mathcal{E}_1] \\ &= \alpha \cdot \sum_{i' \neq p} (\rho_j - v_{i'j}) \frac{x_{i'jp}}{x_{pjp}} \leq \alpha \cdot (v_{pj} - \rho_j). \end{aligned}$$

Above, the second equality follows from linearity and  $\Pr[S_{i'} \ni jp \mid \mathcal{E}_1] = \alpha \cdot \frac{x_{ij'p'}}{x_{pjp}}$ , and the inequality follows from the average-value constraint for bundle  $jp$  (i.e. Equation (4.4)) in our LP. Therefore, by Markov's inequality

$$\Pr \left[ \sum_{i' \in N'_{jp} \setminus \{i\}} (\rho_j - v_{i'j}) > (1 - \beta) \cdot (v_{pj} - \rho_j) \mid \mathcal{E}_1 \right] \leq \frac{\mathbb{E} \left[ \sum_{i' \in N'_{jp} \setminus \{i\}} (\rho_j - v_{i'j}) \mid \mathcal{E}_1 \right]}{(1 - \beta) \cdot (v_{pj} - \rho_j)} \leq \frac{\alpha}{1 - \beta},$$

and thus  $\Pr[\mathcal{E}'_4 \mid \mathcal{E}_1] \geq 1 - \frac{\alpha}{1 - \beta}$ . Recalling that  $\mathcal{E}'_4$  implies  $\mathcal{E}_4$  in any realization, we conclude with the desired bound, as follows.

$$\Pr[\mathcal{E}_4 \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq \Pr[\mathcal{E}'_4 \mid \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = \Pr[\mathcal{E}'_4 \mid \mathcal{E}_1] \geq 1 - \frac{\alpha}{1 - \beta}. \quad \blacktriangleleft$$

Lemma 18 and the preceding discussion yield a lower bound on the probability of an  $N$ -item  $i$  being successfully allocated to a bundle  $jp$  when  $i$ 's deficit is small relative to the excess of the  $P$ -item  $p$ . For the large deficit case, no such bound holds. However, as we now observe (with proof deferred to the full version), large-deficit edges contribute a relative small portion of the allocation's value in the optimal LP solution.

► **Lemma 19.** *Let  $\beta \in [0, 1]$ . For any bundle  $jp$ , let  $L_{jp}^\beta$  denote the set of  $\beta$ -large deficit  $N$ -items for bundle  $jp$ , i.e.,  $N$ -item  $i$  with  $\rho_j - v_{ij} > \beta \cdot (v_{pj} - \rho_j)$ . Then,*

$$\sum_{j,p} \sum_{i \in L_{jp}^\beta} v_{ij} x_{ijp} \leq \frac{1}{\beta} \sum_{j,p} v_{pj} x_{pjp}.$$

## 4.2 Completing the analysis

We are now ready to bound the approximation ratio of Algorithm 1.

► **Theorem 20.** *Algorithm 1 with  $\alpha = 0.3$  is a 32-approximation for AVA.*

**Proof.** Let  $\beta \in [0, 1]$  be some constant to be determined and let  $\gamma = \gamma(\alpha, \beta) := \alpha \cdot (1 - \alpha) \cdot \left(1 - \frac{\alpha}{1 - \beta}\right)$ . Denote  $N_{jp}$  by the set of  $N$ -items allocated to bundle  $jp$  by the algorithm. By Lemmas 17 and 18 we have for bundle  $jp$  and  $N$ -item  $i \notin L_{jp}^\beta$  that

$$\Pr[i \in N_{jp}] = \Pr \left[ \mathcal{E}_4 \mid \bigwedge_{\ell=1}^3 \mathcal{E}_\ell \right] \Pr \left[ \bigwedge_{\ell=1}^3 \mathcal{E}_\ell \right] \geq \left(1 - \frac{\alpha}{1 - \beta}\right) \cdot \alpha \cdot (1 - \alpha) \cdot x_{ijp} = \gamma \cdot x_{ijp}.$$

Therefore, by linearity of expectation and Lemma 19, the expected value of the (feasible) random allocation of Algorithm 1 is at least

$$\begin{aligned} & \sum_{j,p} v_{pj} x_{pjp} + \gamma \sum_{i,j,p:i \neq p} v_{ij} x_{ijp} - \gamma \sum_{j,p} \sum_{i \in L_{jp}^\beta} v_{ij} x_{ijp} \\ & \geq \left(1 - \frac{\gamma}{\beta}\right) \sum_{j,p} v_{pj} x_{pjp} + \gamma \sum_{i,j,p:i \neq p} v_{ij} x_{ijp}. \end{aligned}$$

So, this algorithm's output has value at least a  $\min\{1 - \frac{\gamma}{\beta}, \gamma\}$  fraction of the optimal LP value; i.e., it is a  $1 / \min\{1 - \frac{\gamma}{\beta}, \gamma\}$ -approximation. Taking  $\alpha \approx 0.3$  and  $\beta \approx 0.156$  (optimized by an off-the-shelf numerical solver) yields a ratio of  $1/0.13 < 8$ . The theorem then follows from Lemma 16 and Lemma 10.  $\blacktriangleleft$

### 4.3 Extension: adding side constraints

Before moving on to our online algorithms, we note that the LP-based approach allows us to incorporate additional constraints seamlessly. For example, our LP and algorithm, with minor modifications, allow to approximate allocation problems with both the average-value constraint and  $O(1)$  many budget constraints (for every buyer), corresponding to different resources. More formally, for a cost function  $\ell$  (e.g., corresponding to storage, time, or other costs), each buyer  $j$  has some budget  $B_j^{(\ell)}$ , and the  $\ell$ -cost of allocation to buyer  $j$  must not exceed this budget. That is, for  $x_{ij} \in \{0, 1\}$  an indicator for item  $i$  being allocated to buyer  $j$ , we have

$$\forall j, \quad \ell\text{-cost}_j = \sum_i \ell_{ij} x_{ij} \leq B_j^{(\ell)}. \quad (4.8)$$

The *small-cost* assumption (a.k.a. the small-bids assumption for online AdWords [32]) stipulates that no particular item has high cost compared to the budget, i.e.  $\max_{ij} \ell_{ij}/B_j^{(\ell)} \leq \varepsilon \rightarrow 0$ .

► **Theorem 21.** *There exists a constant-approximate algorithm for AVA and any constant number of budget constraints (for every buyer) subject to the small-bids assumption.*

We defer the proof of the above result to the full version. The same arguments in this section extend to our online algorithms, but are omitted for brevity.

## 5 Online Algorithms: Approximating the Online Optimum

In this section and the next we study AVA in the online i.i.d. setting (see Section 1.3 for definition and notation). Specifically, in this section we provide a polynomial-time online algorithm which provides a constant approximation of the optimal online algorithm.

First, by Lemma 6, we have that the optimal online algorithm is approximated within a factor two by a bundling-based online algorithm which is *committed*. As we will show, the following LP provides a relaxation for the value of the best such online algorithm. Our LP consists of variables  $x_{ijp}$  for each item type  $i \in [m]$ , buyer  $j \in [n]$  and item type  $p$  such that  $(p, j)$  is a  $P$ -edge.

$$\max \sum_{i,j,p} v_{ij} x_{ijp} \quad (\text{OPTon-Bundle-LP})$$

$$\text{s.t.} \quad \sum_i (\rho_j - v_{ij}) x_{ijp} \leq 0 \quad \forall P\text{-edge type } (p, j) \quad (5.9)$$

$$\sum_{j,p} x_{ijp} \leq q_i \cdot T \quad \forall \text{ item type } i \quad (5.10)$$

$$x_{ijp} \leq x_{pjp} \cdot q_i \cdot T \quad \forall N\text{-edge type } (i, j), P\text{-edge type } (p, j) \quad (5.11)$$

$$x_{p'jp} = 0 \quad \forall P\text{-edge types } (p, j) \neq (p', j) \quad (5.12)$$

$$x_{ijp} \geq 0 \quad \forall \text{ item type } i, P\text{-edge type } (p, j)$$

► **Lemma 22.** *(OPTon-Bundle-LP) has value which is at least half the expected value of any online AVA algorithm under i.i.d. arrivals (from the same distribution used in the LP), where item type  $i$  is drawn with probability  $q_i$ .*

**Proof.** First, by the Online Bundling Lemma (Lemma 8), the best committed online bundling-based algorithm 2-approximates the best online algorithm. We therefore turn to showing that (OPTon-Bundle-LP) is a relaxation of the value of the best committed bundling-based online algorithm,  $\mathcal{A}$ . Let  $x_{ijp}$  be the average number of times a copy of item type  $i$  is allocated in a copy of bundle  $jp$  by  $\mathcal{A}$ . Constraint (5.9) follows by linearity of expectation, together with the fact that each opened copy of bundle  $jp$  must satisfy the average-value constraint. Constraint (5.10) simply asserts that  $i$  is allocated at most as many times as it arrives. Constraint (5.11) holds for a committed online algorithm (that guarantees feasibility with probability 1), for the following reason: for every copy of bundle  $jp$  opened, no items can be placed in that bundle before it is opened. But the expected number of copies of  $i$  to be assigned after any bundle  $jp$  is opened is at most the number of arrivals of  $i$  after this bundle is opened and is at most  $q_i \cdot T$ , which upper-bounds the ratio between  $x_{ijp}$  and  $x_{pjp}$ . All other constraints hold similarly to their counterparts in the proof of Lemma 16.  $\blacktriangleleft$

**Note.** Constraint (5.11) is reminiscent of constraints bounding the optimal online algorithm in the secretary problem literature [12] and prophet inequality literature [34].

The outline of our algorithm is similar to that of Algorithm 1, though as it does not have random access to the different items throughout, it first allocates  $P$ -edges in the first  $T/2$  arrivals, and only then allocates  $N$ -edges in the last  $T/2$  arrivals. To distinguish between bundles opened at different times, we now label copies of bundle type  $jp$  (i.e., items allocated to buyer  $j$  with single  $P$ -edge of type  $(p, j)$ ) opened at time  $t$  by  $jpt$ . The algorithm's pseudocode is given in Algorithm 2.

Note that in our online algorithms (here and in Appendix A), the LPs are based on distributions that can be ambiguous in the sense that each item type in the distribution can have both  $P$ -edges and  $N$ -edges, and we don't explicitly modify the distribution to make it unambiguous. However, our algorithm effectively makes each realized instance (of  $T$  sampled items) unambiguous, as we ignore all  $N$ -edges incident to the first  $T/2$  items and vice versa for the last  $T/2$  items.

■ **Algorithm 2** Online rounding of bundling-based LP.

---

```

1: Let  $\mathbf{x}$  be an optimal solution to Equation (OPTon-Bundle-LP)
2: for all arrivals  $t = 1, \dots, T/2$ , of type  $p$  do
3:   Pick a  $j$  according to the distribution  $\{\frac{x_{pjp}}{q_p \cdot T}\}_{j=1, \dots, n}$  and open bundle  $jpt$ 
4: for all arrival  $t^* = T/2 + 1, \dots, T$  of type  $i$  do
5:    $S_{it^*} \leftarrow \emptyset$ 
6:   for all bundles  $jpt$ , with probability  $\frac{\alpha \cdot x_{ijp}}{x_{pjp} \cdot q_i \cdot T}$  do
7:     if bundle  $jpt$  is open then
8:        $S_{it^*} \leftarrow S_{it^*} \cup \{jpt\}$ 
9:   if  $|S_{it^*}| = 1$  then
10:    if  $jpt \in S_{it^*}$  remains permissible after adding  $it^*$  to it then
11:      Allocate  $it^*$  to  $jpt$ 

```

---

## 5.1 Analysis

In what follows we provide a brief overview of the relevant events in the analysis of Algorithm 2, deferring proofs reminiscent of the analysis of Algorithm 1 to the full version.

## 13:16 The Average-Value Allocation Problem

First, the value obtained from  $P$ -edges by Algorithm 2 is clearly half that of the LP, by linearity of expectation. In particular, we create  $x_{pjp}/2$  copies of bundle  $jp$  in expectation. The crux of the analysis is in bounding our gain from  $N$ -edges.

To bound the contribution of  $N$ -edges, we note that a copy of item  $i$  at time  $t^* > T/2$ , which we denote by  $it^*$ , is assigned to bundle  $jpt$  if and only if all the five following events (overloading notation from Section 4) occur:

1.  $\mathcal{E}_0$ : the event that  $it^*$  is the realized item at time  $t^*$ , which happens with probability  $q_i$
2.  $\mathcal{E}_1$ : the event that bundle  $jpt$  is open, which happens with probability  $q_p \cdot \frac{x_{pjp}}{q_p \cdot T} = \frac{x_{pjp}}{T}$ .
3.  $\mathcal{E}_2$ : the event that the Bernoulli( $\frac{\alpha \cdot x_{ijp}}{x_{pjp} \cdot q_i \cdot T}$ ) in Section 5 comes up heads for  $jpt$ .
4.  $\mathcal{E}_3$ : the event that  $S_{it^*} \setminus \bigcup_{j'p'} \bigcup_{t' \neq t} \{j'p't'\} = \emptyset$ .
5.  $\mathcal{E}_4$ : the event that  $jpt$  would remain permissible if we were to add  $it^*$  to bundle  $jpt$ .

Similarly to the events we studied when analyzing our offline Algorithm 1, the events  $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2$  are independent, as are the events  $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ . However,  $\mathcal{E}_3$  is not independent of  $\mathcal{E}_0$  (in particular, it occurs trivially if  $\mathcal{E}_0$  does not). Nonetheless, bounding  $\Pr \left[ \bigwedge_{\ell=0}^3 \mathcal{E}_\ell \right]$  is not too hard. The following lemma, whose proof essentially mirrors that of Lemma 17, and is thus deferred to the full version, provides a bound on the probability of all first four events occurring.

► **Lemma 23.**  $\Pr[\mathcal{E}_0 \wedge \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq \alpha \cdot (1 - \alpha/2) \cdot \frac{x_{ijp}}{T^2}$ .

As with our offline Algorithm 1, the challenge in the analysis is due to possible negative correlations between  $\mathcal{E}_4$  and  $\mathcal{E}_3$ . Similarly, we overcome this challenge of negative correlations, provided  $(i, j)$  has small deficit compared to  $(p, j)$ 's excess, by coupling with an algorithm with no such correlations. (We address large-deficit  $(i, j)$  later.) The obtained syntactic generalization of Lemma 18, whose proof is deferred to the full version, is the following.

► **Lemma 24.** *Let  $\beta \in [0, 1]$ . If  $i, j, p$  are such that  $\rho_j - v_{ij} \leq \beta \cdot (v_{pj} - \rho_j)$ , then*

$$\Pr[\mathcal{E}_4 \mid \mathcal{E}_0 \wedge \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] \geq 1 - \frac{\alpha}{2(1-\beta)}.$$

Lemma 24 and the preceding discussion yield a lower bound on the probability of a copy of item  $i$  be allocated to a bundle  $jpt$  at time  $t^*$  if  $i, j, p$  is in the small deficit case as the above lemma. For large-deficit items, no such bound holds. However, large-deficit edges contribute a small portion of the allocation's value. Specifically, Lemma 19, holds for (OPTon-Bundle-LP) as well, since the only constraint that this lemma's proof relied on was Constraint (4.4), which is identical to Constraint (5.9) in (OPTon-Bundle-LP).

We are now ready to bound the approximation ratio of Algorithm 1.

► **Theorem 25.** *Algorithm 1 with  $\alpha = 0.64$  is a polynomial-time algorithm achieving a 57-approximation of the optimal online algorithm for AVA under known i.i.d. arrivals.*

**Proof.** That the algorithm runs in polynomial time follows from its description, together with the LP (OPTon-Bundle-LP) having polynomial size (in the distribution size). The analysis is essentially identical to that of Theorem 20, with the following differences. First, we recall that the expected number of copies of bundle  $jp$  opened is  $\frac{T}{2} \cdot q_p \cdot \frac{x_{pjp}}{q_p \cdot T} = \frac{1}{2} x_{pjp}$ . Next, by Lemmas 23 and 24, the probability that copy  $it^*$  of small-deficit item  $i$  for bundle  $jpt$  is allocated to it is at least  $\gamma \cdot \frac{x_{ijp}}{T^2}$ , for  $\gamma = \gamma(\alpha, \beta) := \frac{\alpha}{2} \cdot (1 - \frac{\alpha}{2}) \cdot (1 - \frac{\alpha}{2(1-\beta)})$ . Again, linearity of expectation and summation over all  $(t, t^*) \in [T/2] \times (T/2, T]$  in combination with Lemma 19 implies that for any  $\beta \in [0, 1]$ , the gain of Algorithm 2 is at least

$$\begin{aligned} & \frac{1}{2} \left( \sum_{j,p} v_{pj} x_{pj} + \frac{\gamma}{4} \sum_{i,j,p:i \neq p} v_{ij} x_{ij} - \frac{\gamma}{4} \sum_{j,p} \sum_{i \in L_{jp}^\beta} v_{ij} x_{ij} \right) \\ & \geq \left( \left( \frac{1}{2} - \frac{\gamma}{4\beta} \right) \sum_{j,p} v_{pj} x_{pj} + \frac{\gamma}{4} \sum_{i,j,p:i \neq p} v_{ij} x_{ij} \right). \end{aligned}$$

Therefore, by Lemma 22, Algorithm 2 yields a  $2/\min\{\frac{1}{2} - \frac{\gamma}{4\beta}, \frac{\gamma}{4}\}$ -approximation. This expression is optimized by  $\alpha \approx 0.64$  and  $\beta \approx 0.0766$ , yielding a ratio of  $\approx \frac{2}{0.0355} < 57$ , as claimed.  $\blacktriangleleft$

---

## References

- 1 Gagan Aggarwal, Ashwinkumar Badanidiyuru, and Aranyak Mehta. Autobidding with constraints. In *International Conference on Web and Internet Economics*, pages 17–30. Springer, 2019.
- 2 Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1405–1424, 2014.
- 3 Nima Anari, Rad Niazadeh, Amin Saberi, and Ali Shameli. Nearly optimal pricing algorithms for production constrained and laminar bayesian selection. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 91–92, 2019.
- 4 Nick Arnosti and Will Ma. Tight guarantees for static threshold policies in the prophet secretary problem. *Operations Research*, 2022.
- 5 Makis Arsenis and Robert Kleinberg. Individual fairness in prophet inequalities. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, page 245, 2022.
- 6 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.
- 7 Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Naor, et al. Online algorithms for covering and packing problems with convex objectives. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 148–157. IEEE, 2016.
- 8 Santiago Balseiro, Yuan Deng, Jieming Mao, Vahab Mirrokni, and Song Zuo. Robust auction design in the auto-bidding world. *Advances in Neural Information Processing Systems*, 34:17777–17788, 2021.
- 9 Kshipra Bhawalkar, Zhe Feng, Anupam Gupta, Aranyak Mehta, David Wajc, and Di Wang. The average-value allocation problem, 2024. [arXiv:2407.10401](https://arxiv.org/abs/2407.10401).
- 10 Mark Braverman, Mahsa Derakhshan, and Antonio Molina Lovett. Max-weight online stochastic matching: Improved approximations against the online benchmark. In *23rd ACM Conference on economics and Computation*, pages 967–985, 2022.
- 11 Niv Buchbinder, Kamal Jain, and Joseph Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264, 2007.
- 12 Niv Buchbinder, Kamal Jain, and Mohit Singh. Secretary problems via linear programming. *Mathematics of Operations Research*, 39(1):190–206, 2014.
- 13 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.
- 14 Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. *SIAM Journal on Computing (SICOMP)*, 39(6):2189–2211, 2010.

## 13:18 The Average-Value Allocation Problem

- 15 Yuan Deng, Jieming Mao, Vahab Mirrokni, and Song Zuo. Towards efficient auctions in an auto-bidding world. In *Proceedings of the Web Conference 2021, WWW '21*, pages 3965–3973. Association for Computing Machinery, 2021.
- 16 Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *ACM Conference on Electronic Commerce*, pages 71–78, 2009. doi:10.1145/1566374.1566384.
- 17 Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 137–144. ACM, 2012. doi:10.1145/2213977.2213992.
- 18 Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *ACM Conference on Electronic Commerce*, pages 29–38, 2011. doi:10.1145/1993574.1993581.
- 19 Tomer Ezra, Michal Feldman, Nick Gravin, and Zhihao Gavin Tang. “who is next in line?” on the significance of knowing the arrival order in bayesian online settings. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3759–3776, 2023.
- 20 Auto-bidding products support page. <https://www.facebook.com/business/help/1619591734742116>, 2022. Accessed: 2023-07-12.
- 21 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
- 22 Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *ESA (1)*, pages 182–194, 2010. doi:10.1007/978-3-642-15775-2\_16.
- 23 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):416–431, 2011.
- 24 Negin Golrezaei, Ilan Lobel, and Renato Paes Leme. Auction design for roi-constrained buyers. In *Proceedings of the Web Conference 2021, WWW '21*, pages 3941–3952, 2021.
- 25 Auto-bidding products support page. <https://support.google.com/google-ads/answer/2979071>, 2022. Accessed: 2023-07-12.
- 26 Anupam Gupta and Marco Molinaro. How the experts algorithm can help solve LPs online. *Math. Oper. Res.*, 41(4):1404–1431, 2016.
- 27 Johan Håstad. Clique is hard to approximate within  $n^{1-\epsilon}$ . In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 627–636, 1996.
- 28 Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426, 2020.
- 29 Kristen Kessel, Ali Shameli, Amin Saberi, and David Wajc. The stationary prophet inequality problem. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 243–244, 2022.
- 30 Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. *SIAM J. Comput.*, 47(5):1939–1964, 2018.
- 31 Aranyak Mehta. Auction design in an auto-bidding setting: Randomization improves efficiency beyond VCG. In *Proceedings of the ACM Web Conference 2022*, pages 173–181, 2022.
- 32 Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (J.ACM)*, 54(5):22, 2007.
- 33 Joseph (Seffi) Naor, Aravind Srinivasan, and David Wajc. Online dependent rounding schemes. *arXiv preprint arXiv:2301.08680*, 2023.
- 34 Christos Papadimitriou, Tristan Pollner, Amin Saberi, and David Wajc. Online stochastic max-weight bipartite matching: Beyond prophet inequalities. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 763–764, 2021.



- 35 Sebastian Perez-Salazar, Mohit Singh, and Alejandro Toriello. The iid prophet inequality with limited flexibility. *arXiv preprint arXiv:2210.05634*, 2022.
- 36 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

## A Online Algorithms: Approximating the Offline Optimum

In this section we look at the lower and upper bounds of the competitive ratio for online algorithms, i.e. the approximation of the ex-post optimum allocation’s value, and we consider both the adversarial and i.i.d. cases.

**Adversarial arrival.** In this setting, we note that no online algorithm can be  $o(T)$ -competitive. To see this, consider the unit- $\rho$  instance where the first  $T - 1$  arriving items have value  $1 - \varepsilon$  for all  $n = T$  buyers, followed by a single item at the end with value  $1 + \varepsilon T$  for a single adversarially chosen buyer and value 0 for all other buyers. Any online algorithm cannot allocate any of the first  $T - 1$  items due to the average-value constraint, and thus can only get value  $1 + \varepsilon T$  from the last item. In contrast, the ex-post optimum can allocate all items to one buyer and collect value  $T + 1 - \varepsilon$ . On the other hand, a competitive ratio of  $T$  is trivial to achieve for online AVA, by simply allocating any item  $i$  with a  $P$ -edge  $(i, j)$  greedily to the buyer  $j$  yielding the highest value. This is a feasible allocation and has value equal to the highest-valued edge in the  $T$ -item instance, which is obviously at least a  $1/T$  fraction of the optimal allocation’s value.

The rest of this section will therefore be dedicated to AVA with i.i.d. arrivals, as in Section 5, but now focusing on approximating the ex-post optimum. We start with the following result lower bounding the competitive ratio.

► **Lemma 26.** *There exists a family of uniform online i.i.d. unambiguous unit- $\rho$  AVA instances with  $n = m = T \geq 2$  growing, on which every online algorithm’s approximation ratio of the ex-post optimum is at least  $\Omega\left(\frac{\ln n}{\ln \ln n}\right) = \Omega\left(\frac{\ln m}{\ln \ln m}\right) = \Omega\left(\frac{\ln T}{\ln \ln T}\right)$ .*

**Proof.** Let  $\varepsilon = \frac{1}{T}$ . Consider an instance with  $T$  buyers  $j_1, \dots, j_T$ , where all buyers have  $\rho = 1$ , and  $T$  item types. Each item type  $i \in [T - 1]$  is an  $N$ -item, with value  $1 - \varepsilon$  for buyer  $j_i$  and value zero for all others. (So, buyer  $j_T$  has zero value for all  $N$ -items.) The single  $P$ -item type  $T$  has value  $1 + \varepsilon T$  for all buyers. The  $T$  arrival types are drawn uniformly from these  $T$  types, and consequently there is a single arrival of each type in expectation. Now, an online algorithm (that guarantees average-value constraints in any outcome) can only allocate  $N$ -items to a buyer after the buyer was allocated a  $P$ -item. But since each  $N$ -item appears only once in expectation (and hence at most once after the arrival of a  $P$ -item type), each allocation of a  $P$ -item (and  $N$ -items) to a buyer yields expected value at most  $1 + \varepsilon T + 1 - \varepsilon = 3 - \varepsilon$  to an online algorithm. Since only one  $P$ -item arrives in expectation, an online algorithm accrues value at most  $3 - \varepsilon$  in expectation on this instance family.

In contrast, the event  $\mathcal{E}$  that a single  $P$ -item arrived satisfies  $\Pr[\mathcal{E}] = T \cdot \frac{1}{T} \cdot \left(1 - \frac{1}{T}\right)^{T-1} \geq \left(1 - \frac{1}{T}\right)^T \geq \frac{1}{4}$ . Conditioned on  $\mathcal{E}$ , we have a multi-nomial distribution for the number of arrivals  $A_i$ ’s of the  $N$ -item types. Therefore, by standard anti-concentration arguments for the classic balls and bins process [6], we have

$$\Pr\left[\max_i A_i \geq \frac{\ln T}{\ln \ln T} - 1 \mid \mathcal{E}\right] = 1 - o(1).$$

Consequently, the offline algorithm which, if event  $\mathcal{E}$  occurs, allocates the single  $P$ -item and all copies of  $i^* := \arg \max_i A_i$  to  $j_{i^*}$  yields expected value at least  $\mathbb{E}[\max_i A_i \mid \mathcal{E}] \cdot \Pr[\mathcal{E}] = \Omega\left(\frac{\ln T}{\ln \ln T}\right)$ . Consequently, this asymptotic ratio also lower bounds any online algorithm's approximation ratio of the ex-post optimum. The full lemma statement follows, since  $n = m = T$ .  $\blacktriangleleft$

### A.1 A matching algorithm assuming constant expected arrivals

Lemma 26 relied on anti-concentration. If the expected number of arrivals  $A_i$  of each item type  $i$  is at least some constant  $\Gamma > 0$ , namely  $\mathbb{E}[A_i] = q_i \cdot T \geq \Gamma$  (e.g., in Lemma 26 we had  $q_i \cdot T = 1$  for every  $i$ ), then this anti-concentration is tight. In particular, we have the following, by standard Chernoff bounds and union bound (see the full version for proof).

► **Observation 27.** *If  $\mathbb{E}[A_i] \geq \Gamma$  for all  $i \in [m]$  and  $\kappa := \frac{6}{\min(1, \Gamma)} \cdot \frac{\ln T}{\ln \ln T}$ , then*

$$\Pr\left[\max_i A_i \geq \kappa \cdot q_i \cdot T\right] \leq \frac{1}{T^2}.$$

We will show that if the distribution satisfies the assumption on all  $\mathbb{E}[A_i] \geq \Gamma = \Theta(1)$ , we can show an asymptotically matching upper-bound  $O\left(\frac{\ln T}{\ln \ln T}\right)$  of the competitive ratio.

Our first ingredient towards this proof will, naturally, be another LP, this time capturing possible anti-concentration of arrivals. Similar to (OPTon-Bundle-LP), the LP has one variable  $x_{ijp}$  for each item type  $i \in [m]$ , buyer  $j \in [n]$  and item type  $p$  such that  $(p, j)$  is a  $P$ -edge.

$$\max \sum_{i,j,p} v_{ij} x_{ijp} \quad (\text{OPToff-Bundle-LP})$$

$$\text{s.t.} \quad \sum_i (\rho_j - v_{ij}) x_{ijp} \leq 0 \quad \forall P\text{-edge type } (p, j) \quad (\text{A.13})$$

$$\sum_{jp} x_{ijp} \leq 2 \cdot \lceil q_i \cdot T \rceil \quad \forall \text{ item type } i \quad (\text{A.14})$$

$$x_{ijp} \leq x_{pjp} \cdot \lceil q_i \cdot T \cdot \kappa \rceil \quad \forall N\text{-edge type } (i, j), P\text{-edge type } (p, j) \quad (\text{A.15})$$

$$x_{p'jp} = 0 \quad \forall P\text{-edge types } (p, j) \neq (p', j) \quad (\text{A.16})$$

$$x_{ijp} \geq 0 \quad \forall \text{ item type } i, P\text{-edge type } (p, j)$$

► **Lemma 28.** *Fix an AVA instance with i.i.d. arrivals satisfying  $q_i \cdot T \geq \Gamma = \Theta(1)$  for all  $i \in [m]$ . Let OPT be the ex-post optimal value and let  $V[\text{OFF}]$  be the value of (OPToff-Bundle-LP). Then,*

$$\mathbb{E}[\text{OPT}] \leq O(V[\text{OFF}]).$$

**Proof.** By Lemma 16, we can restrict to the optimal ex-post bundling-based solution and just lose a factor of 2 in the approximation ratio. We start with a trivial upper-bound on the value of OPT in any outcome of the i.i.d. arrivals. Consider the instance with exactly one copy of each item type from the support of the distribution. The best bundling-based offline solution for this instance is upper-bounded by (Bundle-LP) (Lemma 16), and this value is clearly upper bounded by  $V[\text{OFF}]$  since the constraints for (Bundle-LP) are tighter than those of (OPToff-Bundle-LP). Under  $T$  i.i.d. arrivals, each item can appear at most  $T$  times, and thus by the Supply Lemma (Lemma 12) applied to the instance with a single occurrence per item type, we find that the following bound holds deterministically.

$$\text{OPT} \leq O(T^2) \cdot V[\text{OFF}].$$

Next, let  $\mathcal{E}$  be the event that no item type  $i$  has more than  $\lceil q_i \cdot T \cdot \kappa \rceil$  arrivals. By Observation 27,  $\Pr[\mathcal{E}] \geq 1 - \frac{1}{T^2}$ . Conditioned on  $\mathcal{E}$ , consider the expected number of times (over the randomness of the i.i.d. arrivals) that the ex-post optimal bundling-based solutions allocate an item of type  $i$  to a copy of bundle  $jp$ , and denote this value by  $x_{ijp}$ . We will argue that such  $x_{ijp}$ 's form a feasible solution for (OPToff-Bundle-LP). Since the expected value of the ex-post optimal bundling-based solution conditioned on  $\mathcal{E}$  is simply  $\sum_{i,j,p} v_{ij} x_{ijp}$ , this immediately gives that  $\mathbb{E}[\text{OPT} \mid \mathcal{E}] \leq 2 \cdot V[\text{OFF}]$ .

The proof that  $x_{ijp}$  constructed above is feasible follows essentially the same argument as Lemma 22. The average-value constraint (A.13) holds by linearity of expectation because the ex-post (bundling-based) optimum for any outcome satisfies the average-value constraint. Constraint (A.14) holds since the expected times we allocate items of type  $i$  cannot exceed  $i$ 's expected number of occurrences, which is bounded by  $\mathbb{E}[A_i \mid \mathcal{E}] \leq \frac{\mathbb{E}[A_i]}{\Pr[\mathcal{E}]} \leq \frac{q_i \cdot T}{1 - 1/T^2} \leq 2 \cdot q_i \cdot T \leq 2 \cdot \lceil q_i \cdot T \rceil$ . Constraint (A.15) holds since whenever a bundle  $jp$  is opened in the ex-post optimum for any outcome, conditioned on  $\mathcal{E}$  we have at most  $q_i \cdot T \cdot \kappa$  items of type  $i$ , which is a trivial upperbound on how many items of type  $i$  can be allocated to bundle  $jp$ , and thus cap the ratio between  $x_{ijp}$  and  $x_{pjp}$ .

Combining the above arguments together with linearity of expectation, the lemma follows.

$$\mathbb{E}[\text{OPT}] = \mathbb{E}[\text{OPT} \mid \mathcal{E}] \cdot \Pr[\mathcal{E}] + \mathbb{E}[\text{OPT} \mid \bar{\mathcal{E}}] \cdot \Pr[\bar{\mathcal{E}}] \leq O(V[\text{OFF}]). \quad \blacktriangleleft$$

We make the simple observation that (OPTon-Bundle-LP) and (OPToff-Bundle-LP) only differ at the RHS of the constraints, with the most crucial difference being in the constraints upper bounding  $x_{ijp}/x_{pjp}$ , where they differ by a factor of  $\frac{\lceil q_i \cdot T \cdot \kappa \rceil}{q_i \cdot T} = O(\kappa)$  (using that  $\Gamma = \Omega(1)$ ). As we prove in the full version, scaling down any feasible solution of the latter LP by  $O(\kappa)$  yields a feasible solution to the former LP, leading to the following observation.

► **Observation 29.** *Fix an AVA instance with i.i.d. arrivals, satisfying  $q_i \cdot T \geq \Gamma = \Theta(1)$  for all item type  $i$ . Then,  $V[\text{OFF}]$  and  $V[\text{ON}]$ , the values of (OPToff-Bundle-LP) and (OPTon-Bundle-LP) (respectively) satisfy  $V[\text{OFF}] \leq O\left(\frac{\ln T}{\ln \ln T}\right) \cdot V[\text{ON}]$*

In our proof of Theorem 25, we showed that Algorithm 2 achieves value at least  $\Omega(V[\text{ON}])$ . Consequently, Lemma 28 and Observation 29 imply the following result.

► **Theorem 30.** *Algorithm 2 is an  $O\left(\frac{\ln T}{\ln \ln T}\right)$ -competitive online algorithm for AVA under  $T$  known i.i.d. arrivals with each item type arriving an expected constant number of times.*

► **Remark 31.** Under the stronger assumption that  $\mathbb{E}[A_i] = q_i \cdot T = \Omega(\ln(mT)/\varepsilon^2)$  for each of the  $m$  item types  $i$  (e.g., if  $T$  grows while the distribution  $\{q_i\}$  remains fixed), the number of arrivals of each item is more concentrated: it is  $\mathbb{E}[A_i] \cdot (1 \pm \varepsilon)$  w.h.p. Consequently, natural extensions of the arguments above, with a smaller blow-up of the RHS of the constraints in (OPTon-Bundle-LP), imply that Algorithm 2's competitive ratio improves to  $O(1)$  in this case.

## **B** Hardness Results

In this section we provide hardness of approximation results for AVA and stark impossibility results for the generalization to GenAVA.

### **B.1** Max-Coverage hardness of AVA

Here we prove that AVA is as hard as the Max-Coverage problem, even if restricted to the unit- $\rho$  case.

## 13:22 The Average-Value Allocation Problem

► **Theorem 32** (Hardness of AVA). *For any constant  $\varepsilon > 0$ , it is NP-hard to approximate AVA to a factor better than  $(\frac{\varepsilon}{\varepsilon-1} + \varepsilon)$  even for unit- $\rho$  instances.*

**Proof.** We give a reduction from “balanced” instances of the MAX-COVERAGE problem. Such an instance consists of a set system with  $n$  elements and  $m$  sets, with each set containing  $n/k$  elements. A classic result of [21] shows that for each  $\delta > 0$ , there exist  $n$  and  $k \leq n\delta$ , such that it is NP-hard to distinguish between the following two cases: (a) there exists a perfect partition, i.e.,  $k$  sets in the set system that cover all  $n$  elements (YES-instances), and (b) no collection of  $k$  sets from the set system cover more than  $n(1 - 1/e + \delta)$  elements (NO-instances). We now define a unit- $\rho$  AVA instance consisting of:

1.  $m$  buyers, where each buyer  $i_S$  corresponds to a set  $S$  in the set system,
2.  $k$  identical *choice items*, which have value  $1 + (\varepsilon/2) \cdot n/k$  for every buyer, and
3.  $n$  distinct *element items*, one for each element  $e$ , which has value  $1 - (\varepsilon/2)$  for the buyers  $i_S$  such that set  $S$  contains element  $e$ , and value zero for the other buyers.

For a YES-instance of MAX-COVERAGE, there is a solution with value  $k + n$ : we can assign both the choice and element items to the buyers corresponding to the  $k$  sets in the perfect partition, thereby getting us value  $n + k$ . (The excess for each choice item can subsidize the deficit for the  $n/k$  element items assigned to that buyer.) On the other hand, for a NO-instance, the  $k$  buyers/sets selected by the choice items can give value  $k$  and also subsidize at most  $n(1 - 1/e + \delta)$  element items with deficit. (No other items with deficit can be chosen.) Setting  $\delta = \varepsilon/2$  means the NO-instances have value at most  $k + n(1 - 1/e + \delta) + n\varepsilon/2 \leq n(1 - 1/e + \varepsilon)$ . This gives a gap between instances with value at least  $n$  and at most  $n(1 - 1/e + \varepsilon)$ , proving the theorem. ◀

### B.2 Clique hardness of GenAVA

Next, we prove that approximating GenAVA defined in (1.2) is as hard as approximating the maximum independent set number in a graph. Recall that the objective in GenAVA is to maximize welfare  $\sum_{ij} v_{ij} x_{ij}$  subject to the more general return-on-spend (ROS) constraints:

$$\forall j, \quad \sum_i v_{ij} x_{ij} \geq \rho_j \cdot \left( \sum_i c_{ij} x_{ij} \right). \quad (\text{B.17})$$

Without loss of generality, we scale  $c_{ij}$  and ensure that all  $\rho_j = 1$ . We show the hardness even for the case where costs depend only on the items, i.e.,  $c_{ij} = c_i$  for each item  $i$ . (The case where  $c_{ij} = c_j$  for each buyer  $j$  is much easier – equivalent to the AVA problem – because we can just fold the  $c_j$  term into the  $\rho_j$  threshold.)

► **Theorem 33** (Hardness of GenAVA). *For any constant  $\varepsilon > 0$ , it is NP-hard to approximate GenAVA for  $n$ -buyer instances with  $\Omega(n^2)$  items to better than a factor of  $n^{1-\varepsilon}$ .*

The proof uses a reduction from the Maximum Independent Set problem. The reduction proceeds as follows: given a graph  $G = (V, E)$  with  $|V| = n$ , define  $M := 2|E|/n^\varepsilon$ , and construct the following GenAVA instance.

1. For each vertex  $v \in V$ , there is a buyer  $j_v$  with  $\rho_{j_v} = 1$ .
2. For each vertex  $v \in V$ , there is a *vertex item*  $i_v$  with item cost  $c_i := M + \deg(v)$ , where  $\deg(v)$  is  $v$ 's degree in  $G$ ; it has value  $M$  for the buyer  $j_v$ , and zero value for all other buyers.
3. For each edge  $e = (u, v) \in E$ , there is an *edge item*  $i_e$  having zero cost; it has value 1 for buyers  $j_u$  and  $j_v$ , and zero value for all others.

**Proof of Theorem 33.** If vertex item  $i_v$  is allocated to buyer  $j_v$ , then by the constraints above, all edge items  $j_e$  with  $e \ni v$  must be allocated to  $i_v$ . Thus, the set of vertices  $U \subseteq V$  whose buyers are sold their respective vertex item is an independent set in  $G$ . Conversely,  $U$  can be taken to be any independent set. Thus, the maximum value obtained by allocating vertex items is precisely  $M \cdot \alpha(G)$ . On the other hand, any optimal allocation must allocate all edge items, as this does not violate any of the ROS constraints. Combining the above, we have that  $OPT = \alpha(G) \cdot M + |E|$ , where  $\alpha(G)$  is the independence number of  $G$ , i.e., the size of the maximum independent set of  $G$ .

Finally, we use the result that for any constant  $\varepsilon > 0$ , it is NP-hard to distinguish between the following two scenarios for an  $n$ -node graph  $G$ : (a)  $G$  contains a clique on  $n^{1-\varepsilon}$  nodes (YES instances), and (b)  $G$  contains no clique on  $n^\varepsilon/2$  nodes (NO instances) [27, 36]. This means that it is NP-hard to distinguish between instances of GenAVA with value at least  $n^{1-\varepsilon} \cdot M$  (corresponding to YES instances) from those with value at most  $(n^\varepsilon/2) \cdot M + |E| = n^\varepsilon \cdot M$  corresponding to the NO instances, and hence proves the claim. ◀

The above hardness construction can, with small changes, show the following hardness results. We defer these additional results' proofs, as well as algorithms showing the (near) tightness of our lower bounds for general GenAVA, to the full version.

► **Theorem 34** (Hardness of i.i.d. GenAVA). *For any constant  $\varepsilon > 0$ , it is NP-hard to  $n^{1-\varepsilon}$ -approximate GenAVA in  $n$ -buyer instances with  $\text{poly}(n)$  items drawn i.i.d. from a known distribution.*

► **Theorem 35** (Hardness of Bicriteria GenAVA). *For any  $\varepsilon > 0$ , it is NP-hard to obtain a solution (which can even be infeasible) to GenAVA that achieves an objective value at least  $\tilde{\Omega}(\sqrt{\varepsilon})$  times the optimal value (i.e. an  $\tilde{O}(1/\sqrt{\varepsilon})$ -approximation), while guaranteeing the cost for each buyer is at most  $1 + \varepsilon$  times their total value, assuming the UGC.<sup>5</sup>*

---

<sup>5</sup> As usual, the soft-Oh notation hides polylogarithmic factors in its argument: i.e.,  $\tilde{O}(f) = f \cdot \text{poly} \log(f)$ .





# Scheduling on a Stochastic Number of Machines

Moritz Buchem  

Technische Universität München, Germany

Franziska Eberle  

Technische Universität Berlin, Germany

Hugo Kooki Kasuya Rosado  

Technische Universität München, Germany

Kevin Schewior  

University of Southern Denmark, Odense, Denmark

Andreas Wiese  

Technische Universität München, Germany

---

## Abstract

We consider a new scheduling problem on parallel identical machines in which the number of machines is initially not known, but it follows a given probability distribution. Only after all jobs are assigned to a given number of bags, the actual number of machines is revealed. Subsequently, the jobs need to be assigned to the machines without splitting the bags. This is the stochastic version of a related problem introduced by Stein and Zhong [SODA 2018, TALG 2020] and it is, for example, motivated by bundling jobs that need to be scheduled by data centers. We present two PTASs for the stochastic setting, computing job-to-bag assignments that (i) minimize the expected maximum machine load and (ii) maximize the expected minimum machine load (like in the Santa Claus problem), respectively. The former result follows by careful enumeration combined with known PTASs. For the latter result, we introduce an intricate dynamic program that we apply to a suitably rounded instance.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** scheduling, approximation algorithms, stochastic machines, makespan, max-min fair allocation, dynamic programming

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.14

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.15737>

**Funding** *Franziska Eberle:* Supported by the Dutch Research Council (NWO), Netherlands Vidi grant 016.Vidi.189.087.

*Kevin Schewior:* Supported by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B.

## 1 Introduction

Stein and Zhong [20] recently introduced scheduling problems in which the number of the given (identical) machines is initially unknown. Specifically, all jobs must be assigned to a given number of bags before the actual number of machines is revealed. When that happens, the bags cannot be split anymore and they have to be assigned to the machines as whole bags, optimizing some objective function. Such problems arise, e.g., when “bundling” jobs to be scheduled in data centers, where the number of available machines depends on external factors such as momentary demand [4, 20].

The aforementioned work (as well as follow-up works [1, 5]) focused on the robustness of a job-to-bag assignment. Specifically, they assumed a worst-case number of machines and compared their solution with the in-hindsight optimum for the respective objective function,



© Moritz Buchem, Franziska Eberle, Hugo Kooki Kasuya Rosado, Kevin Schewior, and Andreas Wiese; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 14; pp. 14:1–14:15



Leibniz International Proceedings in Informatics

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

i.e., a direct job-to-machine assignment without bags. In contrast to this information-theoretic question, we assume that a distribution of the number of machines is known (e.g., from historical data) and aim to *efficiently compute* a job-to-bag assignment that optimizes the objective function *in expectation* – a common formulation of the objective function for stochastic (scheduling) problems [4, 9, 10, 14, 15, 17, 18]. In other words, we use a “fairer” benchmark for our algorithms, allowing us to sidestep the strong lower bounds by [20]. We are the first to study this novel type of scheduling problem, already proposed in [20].

We consider two classic objective functions: minimizing the maximum machine load (makespan) and maximizing the minimum machine load (Santa Claus). Both objectives are well-studied in the deterministic setting, the special case of our problem with one-point distributions, i.e., the distributions in which only one event happens with positive probability. These problems are well understood from a classic approximation perspective: both are known to be strongly NP-hard [7] and both admit Polynomial-Time Approximation Schemes (PTASs) [11, 21], i.e., polynomial-time  $(1 + \varepsilon)$ -approximation algorithms for any  $\varepsilon > 0$ . In this paper, surprisingly, we recover the same state for the stochastic versions by designing a PTAS in both cases. In contrast to the deterministic setting, we require different techniques tailored to each objective function. For the makespan minimization objective, our main technical contribution is the application and analysis of techniques that have previously been used in approximation schemes for *deterministic* scheduling and packing problems. Our approach for the Santa Claus objective is technically much more intriguing and requires the careful set-up of a novel dynamic program (DP) in order to control its size.

Our results are in stark contrast to classic stochastic scheduling problems, where in some cases the currently best known approximation algorithms have distribution-dependent or even linear guarantees [14, 18]. Even for better-understood problems such as load balancing of stochastic jobs on deterministic machines, previous approaches [4, 9, 15] rely on concentration bounds which inherently prohibit approximation ratios of  $1 + \varepsilon$  for arbitrarily small  $\varepsilon > 0$ . Moreover, PTASs for stochastic load balancing on deterministic machines are only known for identical machines and Poisson distributed jobs [2, 13]. We hope that our positive results inspire research for other scheduling problems with a stochastic number of machines, even for (in the classic model with jobs with stochastic processing times) notoriously hard objective functions such as expected weighted sum of completion times.

## 1.1 Our Contribution and Techniques

Our first result is the following.

► **Theorem 1.** *There is a PTAS for the problem of computing the job-to-bag assignment that minimizes the expected maximum machine load.*

We first guess the bag sizes of the optimal solution up to a factor of  $1 + \varepsilon$ . For each guess, we check whether there is a corresponding assignment of the jobs to the bags (up to a factor of  $1 + \varepsilon$ ), using the PTAS for bin packing with variable sizes [12]. Among the guesses that fulfill this condition, we can select the (approximately) best guess using the PTAS for makespan minimization [11].

For this approach to yield a PTAS, we need to bound the number of guesses by a polynomial (in the input length). First note that it is straightforward to get down to a *quasi-polynomial* number of guesses (and thus a QPTAS). The approach is to disregard jobs of size  $(\varepsilon/n) \cdot p_{\max}$  where  $p_{\max}$  is the largest processing time; indeed, for any solution, such jobs make up at most an  $\varepsilon$ -fraction of the objective-function value. The resulting number of possible guesses for a single bag size is then logarithmic in  $n$ , leading to a quasi-polynomial



number of guesses for the (multi-)set of bag sizes. To get a *polynomial* bound (and thus a PTAS), we make the following crucial observation. Let  $C$  be an estimate for the largest *bag* size, up to a constant factor. While bags of size  $\mathcal{O}(\varepsilon C)$  cannot be disregarded, it is enough to know their *number* rather than approximate size. Intuitively, when computing a bag-to-machine assignment, these bags are treated like “sand”, i.e., as infinitesimal jobs of total volume equal to the total volume of those bags. The number of possible (rounded) bag sizes is hence constant, leading to a polynomial number of guesses for the (multi-)set of bag sizes.

Our second result is significantly harder to achieve.

► **Theorem 2.** *There is a PTAS for the problem of computing the job-to-bag assignment that maximizes the expected minimum machine load.*

One may be tempted to try a similar approach as for our first result. Even getting a QPTAS is, however, not possible in the same way as one cannot simply disregard jobs of size  $(\varepsilon/n) \cdot p_{\max}$ . Consider an instance in which the number of machines is deterministically  $M$  and in which there are one job of size 1 and  $M - 1$  jobs of size  $\varepsilon/(2M)$ . Here, ignoring the jobs of size  $\varepsilon/(2M)$  leads to  $M - 1$  empty bags, yielding an objective function value of 0 instead of the optimal value  $\varepsilon/(2M)$ .

Also, it is no longer true either that for bags of size  $\mathcal{O}(\varepsilon C)$  (where  $C$  is the size of a largest bag) it is enough to know their total number: Consider an instance with optimal objective-function value  $\text{OPT}$  and add one huge job of size  $\text{OPT}/\varepsilon^2$  to the set of jobs and one machine to each scenario. Clearly, this new instance has maximum bag size  $C \geq \text{OPT}/\varepsilon^2$  while the optimal objective-function value does not change since this huge job can safely be packed in its private bag and scheduled on its private machine. (However, crucially for the PTAS for makespan minimization, adding this huge job there would change the objective-function value.) In this example, the probability that scenarios with optimal objective-function value much smaller than  $\varepsilon C$  occur is 1. To obtain a  $(1 + \varepsilon)$ -approximation, however, one still has to compute a  $(1 + \varepsilon)$ -approximation for the original instance, for which the sizes of bags of size  $\mathcal{O}(\varepsilon C)$  are relevant. Of course, the issue with this particular instance could be avoided by removing the huge job in a pre-processing step. However, by concatenating the above original instance at super-constantly many different scales, one can create a new instance where one essentially has to identify the “relevant scales” in a preprocessing step.

In some sense, the first step of proving Theorem 2 is addressing precisely the problem of identifying the correct scales: We show that, at a loss of  $1 + \mathcal{O}(\varepsilon)$  in the approximation guarantee, the problem can be reduced to the case of polynomially bounded processing times that are all powers of  $1 + \varepsilon$ . To do so, we define suitable (non-trivial) subproblems and assemble them to a global solution with a dynamic program (DP). This approach can be seen as a simpler version of our approach for polynomially bounded processing times, which we focus on in the following.

Our general approach is to divide the range of possible bag sizes into intervals that contain  $\mathcal{O}_\varepsilon(1)$  possible approximate bag sizes each. For each such interval, it is then possible to guess the set of bags of the respective size in polynomial time. Considering the same range for sizes of *jobs*, rather than bags, we would also be able to guess the assignment of these jobs to these bags. Observe that a job may, of course, be assigned to a bag whose size lies in a different interval than the job’s size. However, we argue that the precise assignment is only relevant when these intervals are neighboring intervals and, hence, can be guessed in polynomial time. If this is not the case, i.e., if jobs are assigned to a bag whose size lies in a much larger interval, then such jobs are sufficiently small for us to only consider their total volume. The resulting parameters, such as number of bags created so far and the assignment of

smaller jobs to larger intervals, through which the subproblems corresponding to the intervals interact, are kept track of by a DP. While it is straightforward to keep track of all bags from larger intervals as well as the assignment of jobs from these intervals to the bags, it is not clear how to do this with a polynomial-time DP. In particular, when we consider bag sizes of one interval, we still need to remember previously defined bags of much larger sizes with a super-constant number of possibilities for these sizes. In fact, we show that it is sufficient to keep track of a constant number of parameters that capture all necessary information about larger intervals. Using that the processing times are polynomially bounded, we can bound the size of the DP table by a polynomial in the encoding of the input.

When considering a DP cell corresponding to some interval and guessing bag sizes along with the other parameters implied by the discussion above, we need to evaluate the quality of this guess. To do so, we guess additional parameters (also kept track of by the DP). The main observation is as follows. Suppose that, in addition to the aforementioned parameters, we know the relevant range of the number of machines and the bag sizes from the next-lower interval. For each number of machines in the aforementioned range, we assign each bag

- (i) from a higher interval to one machine each,
- (ii) from the two currently relevant intervals optimally, and
- (iii) from the lower intervals fractionally (the total volume of these bags can be approximated).

The reason we may do so is that the bags assigned in (i) are large enough to assign enough load to an entire machine and the bags assigned in (iii) are small enough to be considered fractional.

We remark that for both problems considered, a PTAS is the best possible approximation algorithm achievable when the number of bags (and machines) is part of the input, unless  $P = NP$ : Since their strongly NP-hard deterministic counterparts [8] are special cases of the stochastic problems, neither makespan minimization nor Santa Claus on stochastic machines admits fully polynomial time approximation schemes (FPTASs) unless  $P = NP$ . If, however, the number of bags (and machines) is not part of the input, i.e., a constant, a FPTAS can be designed by directly guessing the bag sizes approximately, i.e., up to a factor of  $1 + \varepsilon$ , in polynomial time and using known FPTASs to compute a job-to-bag assignment based on these bag sizes [6, 16].

Stein and Zhong [20] also considered a third objective function, minimizing the difference between the maximum and the minimum machine load. Any polynomial-time approximation algorithm (in the multiplicative sense) is, however, impossible here unless  $P = NP$ . Indeed, already in the deterministic case, it is strongly NP-hard to decide whether the optimal objective-function value is 0 (as can be seen, e.g., by a straightforward reduction from 3-Partition).

## 1.2 Further Related Work

We first review the literature on the aforementioned information-theoretic question in which one compares with the in-hindsight optimum. Since this benchmark is stronger than ours and the upper bounds are obtained through polynomial-time algorithms, the upper bounds carry over to our setting as guarantees of polynomial-time approximation algorithms. Specifically, for makespan, Stein and Zhong [20] showed how to compute for any  $\varepsilon > 0$  a job-to-bag assignment whose cost is guaranteed to be a factor of at most  $5/3 + \varepsilon$  away from the cost of the in-hindsight optimum. They also showed an impossibility of  $4/3$ . When all jobs have infinitesimal size, the best-possible guarantee is  $(1 + \sqrt{2})/2 \approx 1.207$  [5, 20]. For Santa Claus and infinitesimal jobs, the best-possible guarantee is  $2 \ln 2 \approx 1.386$  [20].

This model has been generalized in two directions. First, Eberle et al. [5] considered arbitrary machine *speeds* (rather than just 0 or 1) that are revealed after the bags have been created. They gave a guarantee of  $2 - 1/m$  with respect to the in-hindsight optimum and improved guarantees for special cases. Second, Balkanski et al. [1] considered the problem with arbitrary speeds in the algorithms-with-predictions framework.

In the majority of the scheduling literature, stochastic uncertainty refers to uncertainty in the processing times of the jobs (see [17] for a survey and [4, 9, 10] for some recent works). The literature on stochastic uncertainty, even uncertainty in general, in the machines is much more scattered. Stadje considered the unrecoverable breakdown on a single machine caused by stochastic jobs [19]. Temporary machine unavailability has also been studied in [3].

## 2 Preliminaries

Formally, we are given a job set  $J = [n] := \{1, \dots, n\}$ , where each job has a *processing time*  $p_j$ , and a maximum number of machines  $M$ . For each  $1 \leq m \leq M$ , we are given its *probability*  $q_m$ , where  $\sum_{m=1}^M q_m = 1$ . We want to find a partition of the job set  $J$  into  $M$  sets, called *bags*. We denote the set of bags by  $\mathcal{B}$ . For a bag  $B \in \mathcal{B}$ , let  $p(B) = \sum_{j \in B} p_j$  denote its *size*. We typically say for  $j \in B$  that  $j$  is *packed* in bag  $B$ .

Clearly, if  $M \geq n$ , we can pack every job in its own private bag, and the problem becomes trivial. Hence, we assume from now on that  $M < n$ .

We denote by  $\text{OPT}(\mathcal{B}, m)$  the optimal objective function value for a given set of bags  $\mathcal{B}$  and a scenario with  $m$  machines, that is,  $\text{OPT}(\mathcal{B}, m)$  denotes the maximum or minimum machine load of an optimal bag-to-machine assignment, or *schedule*, respectively. The objective is to find a partition or set of bags  $\mathcal{B}$  that optimizes  $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m)$ . We denote a fixed optimal set of bags by  $\mathcal{B}^*$  and its objective function value by  $\text{OPT} := \sum_{m=1}^M q_m \text{OPT}(\mathcal{B}^*, m)$ .

As discussed above, the problems we consider are generalizations of strongly NP-hard problems. Thus, unless  $P = NP$ , we cannot expect to find  $\mathcal{B}^*$  in polynomial time. Hence, we are interested in *polynomial-time approximation schemes* (PTASs), i.e., for each  $\varepsilon > 0$ , a polynomial-time  $(1 + \varepsilon)$ -*approximation algorithm*. Such an algorithm is required to return a partition of the job set  $J$  into  $M$  bags, denoted by  $\mathcal{B}$ , that satisfies  $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m) \leq (1 + \varepsilon) \text{OPT}$  for makespan, and  $\sum_{m=1}^M q_m \text{OPT}(\mathcal{B}, m) \geq \frac{1}{1 + \varepsilon} \text{OPT}$  for Santa Claus.

## 3 Minimizing the maximum machine load

In this section we design and analyze our polynomial-time approximation scheme for the setting of makespan minimization: For a given number of machines  $m$  and a set  $\mathcal{B}$  of bags, we want to find an assignment of bags to machines that minimizes the maximum total size of bags assigned to any machine.

### 3.1 Algorithm

Let  $\varepsilon > 0$ ; we will give a polynomial-time algorithm that achieves an approximation ratio of  $1 + \mathcal{O}(\varepsilon)$ . This algorithm finds a good estimate of the optimal bag sizes in  $\mathcal{B}^*$ . To this end, we show later that the maximum size of a bag in  $\mathcal{B}^*$  is at most  $4C$ , where

$$C := \sum_{m=1}^M q_m \max \left\{ \max_{j \in J} p_j, \frac{1}{m} \sum_{j \in J} p_j \right\}.$$

We say a bag  $B$  in  $\mathcal{B}^*$  is *regular* if its size is at least  $\varepsilon C$  or if there is at least one job of size at least  $\varepsilon^2 C$  packed in  $B$ . For  $\ell \in \mathcal{L} := \{\lfloor \log_{1+\varepsilon}(\varepsilon^2 C) \rfloor, \lfloor \log_{1+\varepsilon}(\varepsilon^2 C) \rfloor + 1, \dots, \lfloor \log_{1+\varepsilon}(4C) \rfloor\}$ , the algorithm *guesses* the number  $M_\ell$  of optimal bags with  $p(B) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$ .

Further, it *enumerates* all possible numbers  $M_{\text{sand}}$  of bags of size at most  $(1+\varepsilon)\varepsilon C$ , called *sand bags*. These sand bags do not directly correspond to optimal bags, but instead can pack all jobs not packed in regular bags in OPT.

Clearly, a guess  $(M_\ell)_{\ell \in \mathcal{L}}$  combined with  $M_{\text{sand}}$  sand bags does not necessarily guarantee that it is *feasible*, i.e., that  $M_{\text{sand}} + \sum_{\ell \in \mathcal{L}} M_\ell \leq M$  and that there is a partition of  $J$  into bags such that there are at most  $M_\ell$  bags with sizes in  $[(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$  and  $M_{\text{sand}}$  bags of size at most  $(1+\varepsilon)\varepsilon C$ . Thus, the algorithm ignores all combinations of  $(M_\ell)_{\ell \in \mathcal{L}}$  and  $M_{\text{sand}}$  with more than  $M$  bags. If the total number of bags is at most  $M$ , the algorithm uses the PTAS by Hochbaum and Shmoys [12] for bin packing with variable bin sizes to check if there is a feasible packing of jobs into the bags as follows: The input is  $\varepsilon$  as approximation parameter, an item of size  $p_j$  for each job  $j \in J$ ,  $M_\ell$  bins of size  $(1+\varepsilon)^{\ell+1}$  for any  $\ell \in \mathcal{L}$ , and  $M_{\text{sand}}$  bins of size  $(1+\varepsilon)\varepsilon C$ . If the guess is feasible, the PTAS is guaranteed to return an item-to-bin (here a job-to-bag) assignment that violates the bin sizes by at most a factor  $(1+\varepsilon)$ .

If all jobs can be packed by the above PTAS, the algorithm evaluates the current guess by computing a  $(1+\varepsilon)$ -approximation of  $\text{OPT}((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$ , where we overload notation and let  $\text{OPT}((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$  denote the minimum makespan for a set of bags consisting of  $M_\ell$  bags of size  $(1+\varepsilon)^{\ell+1}$ , for any  $\ell \in \mathcal{L}$ , and  $M_{\text{sand}}$  bags of size  $(1+\varepsilon)\varepsilon C$ . We denote the makespan of this  $(1+\varepsilon)$ -approximation by  $z((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$  and compute it by running the PTAS by Hochbaum and Shmoys [11] for makespan minimization on identical machines with approximation parameter  $\varepsilon$ ,  $M_\ell$  jobs with processing time  $(1+\varepsilon)^{\ell+1}$ ,  $M_{\text{sand}}$  jobs with processing time  $(1+\varepsilon)\varepsilon C$ , and  $m$  machines.

The algorithm returns a feasible minimizer of  $\sum_{m=1}^M q_m z((M_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}, m)$ .

### 3.2 Analysis

In this section, we analyze the algorithm designed in the previous section. We start by justifying our bound on the maximum bag size before we argue that there exists a guess that is similar to the optimal set  $\mathcal{B}^*$  in terms of the bag size and objective-function value. Last, we evaluate the running time of the algorithm and conclude with the proof of Theorem 1. For formal proofs we refer to the full version.

We begin by justifying our assumption to only consider bags of size at most  $4C = 4 \sum_{m=1}^M q_m \max \{ \max_{j \in J} p_j, \frac{1}{m} \sum_{j \in J} p_j \}$ : By [20],  $4C$  is an upper bound on OPT. As the largest bag size lower bounds  $\text{OPT}(\mathcal{B}, m)$  in scenario  $m$ , this implies the next lemma.

► **Lemma 3.** *No optimal solution uses bags of size greater than  $4C$ .*

Fix a set of bags  $\mathcal{B}^*$  with objective-function value OPT. By Lemma 3, the maximum bag size is at most  $4C$ . The algorithm guesses a set of bag sizes similar to the bag sizes in  $\mathcal{B}^*$ .

Based on  $\mathcal{B}^*$  we define a “good” guess  $(\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$ , i.e., a set of possible bag sizes, as follows: Let  $\mathcal{B}_R^*$  denote the set of regular bags, i.e., the set of bags in  $\mathcal{B}^*$  that pack at least one job of size at least  $\varepsilon^2 C$  or have size at least  $\varepsilon C$ .

- For  $\ell \in \mathcal{L}$ , let  $\hat{M}_\ell$  be the number of regular bags in  $\mathcal{B}_R^*$  with  $p(B) \in [(1+\varepsilon)^\ell, (1+\varepsilon)^{\ell+1})$ .
- Set  $\hat{M}_{\text{sand}} = \left\lceil \frac{\sum_{B \in \mathcal{B}^* \setminus \mathcal{B}_R^*} p(B)}{\varepsilon C} \right\rceil$ ; recall that sand bags have size at most  $(1+\varepsilon)\varepsilon C$ .

Since the sizes of regular bags are only rounded up, the bags in  $(\hat{M}_\ell)_{\ell \in \mathcal{L}}$  can pack the same subset of jobs as  $\mathcal{B}_R^*$ . Since the volume of any sand bag has been increased by a  $(1+\varepsilon)$ -factor as opposed to  $\varepsilon C$  and the size of any job not packed in a regular bag is at most  $\varepsilon^2 C$ , we show that the bag-size vector  $\hat{M} := (\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$  is a possible and feasible guess.

► **Lemma 4.** *The above defined vector  $\hat{\mathcal{M}}$  is a feasible guess of the algorithm. The jobs can be packed into a set of bags consisting of  $\hat{M}_\ell$  bags with size  $(1 + \varepsilon)^{\ell+1}$  for  $\ell \in \mathcal{L}$  and  $\hat{M}_{\text{sand}}$  bags with size  $(1 + \varepsilon)\varepsilon C$ .*

Combining this lemma with Theorem 2 of [12], we get the next corollary.

► **Corollary 5.** *The PTAS by [12] returns a packing of all jobs into a set of bags with  $M_\ell$  bags of size  $(1 + \varepsilon)^{\ell+2}$  for  $\ell \in \mathcal{L}$  and  $M_{\text{sand}}$  bags of size  $(1 + \varepsilon)^2\varepsilon C$ .*

To prove the next lemma, we first assign the regular bags in the same way as in the optimal solution and assign the sand bags one by one to the currently least loaded machine. For bounding the makespan in a given scenario  $m$ , we distinguish whether a regular bag determines the makespan (which increases the makespan by at most a factor  $(1 + \varepsilon)$  compared to  $\text{OPT}(\mathcal{B}^*, m)$ ) or whether a sand bag determines the makespan. In the latter case, we use a volume bound to upper bound this sand bag's starting time (losing at most a factor  $(1 + \varepsilon)$ ) and amortize its maximum size, i.e.,  $(1 + \varepsilon)\varepsilon C$ , over all scenarios, using that  $\sum_{m=1}^M q_m = 1$ .

► **Lemma 6.**  *$\hat{\mathcal{M}} = (\hat{M}_\ell)_{\ell \in \mathcal{L} \cup \{\text{sand}\}}$  satisfies  $\sum_{m=1}^M q_m \text{OPT}(\hat{\mathcal{M}}, m) \leq (1 + 5\varepsilon)\text{OPT}$ .*

**Proof of Theorem 1.** Using that we return the cheapest guess and that the number of distinct rounded sizes of regular bags is bounded by  $\mathcal{O}(\frac{1}{\varepsilon^2})$ , we can show the following two lemmas. Combined, they complete the proof of Theorem 1.

► **Lemma 7.** *The set of bags returned by the algorithm guarantees an objective function value of at most  $(1 + \mathcal{O}(\varepsilon))\text{OPT}$ .*

► **Lemma 8.** *For  $\varepsilon \in (0, \frac{1}{2})$ , the algorithm runs in time  $\mathcal{O}\left(\left(\frac{n}{\varepsilon}\right)^{\mathcal{O}(1/\varepsilon^2)}\right)$ .*

## 4 Maximizing the minimum machine load

In this section, we present our polynomial-time  $(1 + \varepsilon)$ -approximation algorithm for the setting in which we want to maximize the minimum machine load. We refer to the full version for the formal proofs for the results presented in this section.

### Polynomially bounded processing times

First, we show that we can reduce our problem to the case of polynomially bounded job processing times that are all essentially powers of  $1 + \varepsilon$ , while losing at most a factor of  $1 + \mathcal{O}(\varepsilon)$  in our approximation guarantee. The main concepts of the reduction can be summarized by the following three ideas.

The first idea is to disregard scenarios whose contribution to the expected objective function value is very small. W.l.o.g., assume that  $p_j \in \mathbb{N}$  and let  $d$  be an integer such that  $\text{OPT}(\mathcal{B}^*, m)$  falls in the interval  $[1, (\frac{n}{\varepsilon})^d]$  for every scenario  $m$ . Then, for some offset  $a \in \{0, 1, \dots, \frac{1}{\varepsilon} + 3\}$  we “split” the interval  $[1, (\frac{n}{\varepsilon})^d]$  into a polynomial number of pairwise disjoint intervals  $\tilde{I}_i = \left[ \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a}, \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i}{\varepsilon} + a} \right)$ . Observe that any two consecutive intervals have a multiplicative gap of  $\left(\frac{n}{\varepsilon}\right)^3$ . Using probabilistic arguments, we show that there is an offset  $a$  such that the scenarios with  $\text{OPT}(\mathcal{B}^*, m)$  in the gaps contribute very little to the expected objective function value. Hence, such scenarios can be neglected by losing a factor of at most  $1 + \mathcal{O}(\varepsilon)$  in the approximation ratio. As there is only a polynomial number of possible offsets  $a$ , we may assume that we correctly choose such  $a$  by enumeration.

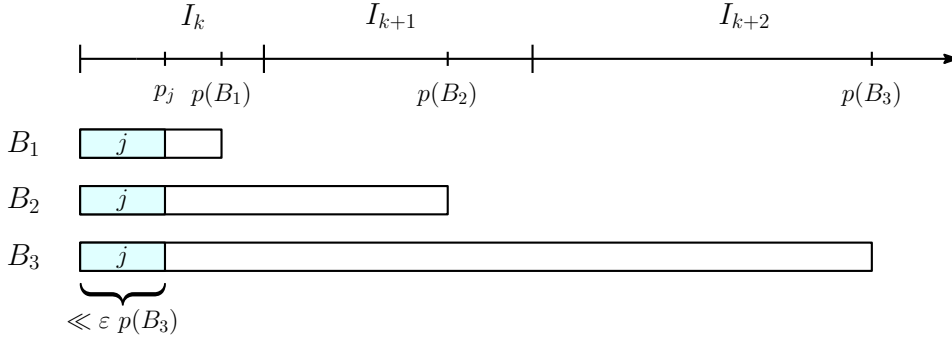
The second idea is to observe that the gaps enable us to actually *ignore* a carefully chosen subset of jobs. Let  $\tilde{I}_i^+ = \left[ \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a - 3}, \left(\frac{n}{\varepsilon}\right)^{3i + \frac{i-1}{\varepsilon} + a} \right) \cup \tilde{I}_i$  denote the extended interval obtained by the union of the interval  $\tilde{I}_i$  and the smaller of its adjacent gaps, and let  $m$  be a scenario such that  $\text{OPT}(\mathcal{B}^*, m) \in \tilde{I}_i$ . We show that, by losing a factor of at most  $1 + \mathcal{O}(\varepsilon)$  in the approximation ratio, we may assume that a machine with minimum load in the schedule that achieves  $\text{OPT}(\mathcal{B}^*, m)$  is assigned only bags with jobs whose processing time is in  $\tilde{I}_i^+$ . Then, based on this assumption, we show that we may assume that there are no jobs whose processing times fall in the gaps by losing at most another factor of  $1 + \mathcal{O}(\varepsilon)$  in the approximation ratio. Observe that we are now facing an instance where neither  $\text{OPT}(\mathcal{B}^*, m)$  nor  $p_j$  belong to the just created gaps in the interval  $\left[1, \left(\frac{n}{\varepsilon}\right)^d\right]$ .

The third idea is then to solve the problem restricted to the intervals  $\tilde{I}_i$  individually by using the fact that within each interval  $\tilde{I}_i$  the processing times are polynomially bounded and combine the obtained solutions into a single one with a dynamic program. We show that rounding up the processing times and solving each subproblem that arises in the dynamic program costs a factor of at most  $1 + \mathcal{O}(\varepsilon)$  in the approximation ratio. Formalizing this proof sketch proves Lemma 9.

► **Lemma 9.** *By losing a factor of at most  $1 + \mathcal{O}(\varepsilon)$  in the approximation ratio, we can assume for each job  $j \in J$  that  $p_j = \lceil (1 + \varepsilon)^{k_j} \rceil$  for some  $k_j \in \mathbb{N}_0$  and  $p_j \in [1, n^{c(\varepsilon)}]$  where  $c(\varepsilon)$  is some global constant.*

**Algorithmic overview.** Based on Lemma 9, we assume that each job  $j \in J$  satisfies  $p_j \in [1, n^{c(\varepsilon)}]$ . The high-level idea of our algorithm is to partition  $[1, n^{c(\varepsilon)}]$  into intervals of the form  $I_k := \left[\left(\frac{1}{\varepsilon}\right)^{3k}, \left(\frac{1}{\varepsilon}\right)^{3k+3}\right)$  for  $k \in \mathbb{N}$  and only consider bags, jobs, and scenarios relevant for a *single* interval. More precisely, we use these intervals to partition the processing times  $\{p_j\}_{j \in J}$ , the bag sizes in  $\mathcal{B}^*$  and in our solution as well as the values  $\{\text{OPT}(\mathcal{B}^*, m)\}_m$ . Let  $K$  such that  $\sum_{j \in J} p_j \in I_K$ ; we ignore intervals  $I_k$  with  $k > K$ . For  $k \in [K]$ , let  $J_k := \{j \in J : p_j \in I_k\}$ , let  $L_k := \{\ell \in \mathbb{N} : \lceil (1 + \varepsilon)^\ell \rceil \in I_k\}$ , and let  $\mathcal{B}_k^* := \{B \in \mathcal{B}^* : p(B) \in I_k\}$ .

Our algorithm recursively considers the intervals in the order  $I_K, I_{K-1}, \dots, I_1$  and, step by step, defines bags that correspond to  $\mathcal{B}_K^*, \mathcal{B}_{K-1}^*, \dots, \mathcal{B}_1^*$ . When considering interval  $I_k$ , the algorithm enumerates all possible bag sizes of bags in  $\mathcal{B}_k^*$  and all possible assignments of a subset of the jobs in  $J_k \cup J_{k-1}$  to those bags; the remaining jobs in  $J_k$  are implicitly assigned to bags in  $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}^*$ . Here, we use the fact that by definition of our intervals only a constant number of jobs in  $J_k \cup J_{k-1}$  can be assigned to any bag in  $\mathcal{B}_k^*$  while jobs in  $J_k$  are tiny compared to bags in  $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$  (see Figure 1 for visualization) and, hence, the assignment of jobs  $J_k$  to bags  $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$  cannot be guessed in polynomial time. The remaining jobs in  $J_{k-1}$  will be assigned when interval  $I_{k-1}$  is considered. We embed this recursion into a polynomial-time dynamic program (DP). Since our DP is quite technical, we first describe the algorithmic steps that correspond to the root subproblem of the recursion, i.e.,  $I_K$ , before we define the DP cells and argue about their solution. Defining the DP cells and solving their corresponding subproblem involves enumerating all possible values of several quantities and storing an approximation of the objective-function value of the (approximately) best combination in the DP cell. When arguing about the correctness of our DP, we show that there is a chain of DP cells that represent some (fixed) optimal solution. Hence, we use  $X^*$  for some parameter  $X$  when referring to the correct value, i.e., the value of this parameter in this optimal solution. We refer to this process as *guessing*  $X^*$ . In general, we use  $\hat{X}$  to refer to an arbitrary guess.



■ **Figure 1** Visualization of relation between jobs in  $J_k$  and bags in  $\mathcal{B}_k^*$ ,  $\mathcal{B}_{k+1}^*$  and  $\bigcup_{k'=k+2}^K \mathcal{B}_{k'}^*$ .

## 4.1 Guessing initial quantities

In the following, we describe how to guess and evaluate initial parameters corresponding to a (partial) solution of our root subproblem. Intuitively, we construct a partial assignment of jobs to bags  $\mathcal{B}_K^*$  and the parameters representing this partial assignment define the DP cell corresponding to the remaining problem.

### 4.1.1 Algorithm

The algorithm to compute a partial solution to a root subproblem can be essentially split into two phases: (1) *guessing* key quantities and (2) *evaluating* these guesses.

**Guessing phase.** We start by guessing  $|\mathcal{B}_K^*|$  and  $|\mathcal{B}_{K-1}^*|$ , the number of bags in  $\mathcal{B}_K^*$  and in  $\mathcal{B}_{K-1}^*$ , before we guess  $(1 + \varepsilon)$ -approximations for the bags sizes in  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ . Formally, for each bag  $B \in \mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$  we guess a value  $\ell(B) \in \mathbb{N}$  such that  $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1})$ ; we say that such a value  $\ell(B)$  is the *size-estimate* for  $B$ . Next, we guess an assignment of all jobs in  $J_K$  and a subset of the jobs in  $J_{K-1}$  to the bags  $\mathcal{B}_K^*$  and an assignment of the remaining jobs in  $J_{K-1}$  and of a subset of the jobs in  $J_{K-2}$  to the bags  $\mathcal{B}_{K-1}^*$ . Finally, we guess  $m_{\max}^{(K)}$  which we define to be the largest value  $m \in M$  for which  $\text{OPT}(\mathcal{B}^*, m) \in I_K$ .

**Evaluation phase.** In contrast to the previous section, maximizing the minimum machine load asks for “covering” a machine or, in our case, a bag. To this end, we potentially have to assign jobs from  $\bigcup_{k'=1}^{K-2} J_{k'}$  to the bags in  $\mathcal{B}_K^*$ . Formally, for  $B \in \mathcal{B}_K^*$  let  $p^+(B)$  be the total size of the jobs from  $J_K$  and  $J_{K-1}$  already assigned to  $B$ . We define  $S := \sum_{B \in \mathcal{B}_K^*} \max\{[(1 + \varepsilon)^{\ell(B)}] - p^+(B), 0\}$ . Our DP also stores this value in order to guarantee that, in the remainder, jobs from  $\bigcup_{k'=1}^{K-2} J_{k'}$  with total size  $S$  are assigned to bags in  $\mathcal{B}_K^*$ . Let  $J_{K-1}(\mathcal{B}_{K-1}^*)$  and  $J_{K-2}(\mathcal{B}_{K-1}^*)$  be the subsets of  $J_{K-1}$  and  $J_{K-2}$  already assigned to bags in  $\mathcal{B}_{K-1}^*$ . Then,  $\bar{S} := \sum_{B \in \mathcal{B}_{K-1}^*} [(1 + \varepsilon)^{\ell(B)}] - p(J_{K-1}(\mathcal{B}_{K-1}^*) \cup J_{K-2}(\mathcal{B}_{K-1}^*))$  is the total volume of bags in  $\mathcal{B}_{K-1}^*$  that needs to be covered with jobs from  $\bigcup_{k'=1}^{K-3} J_{k'}$ .

For evaluating our current guess, we fix some  $m \leq m_{\max}^{(K)}$  and create a set  $J_T$  of dummy jobs, each with processing time 1 and total size  $T := \sum_{k=1}^{K-2} \sum_{j \in J_k} p_j - S - \bar{S}$ . Now, we guess the assignment of the bags  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$  to the machines. Based on the load guaranteed by these bags, we now greedily distribute these dummy jobs as follows. Assume w.l.o.g. that the machines are sorted non-decreasingly by their loads and consider the prefix of the machines which all have the smallest load. We assign to each of these machines the same

number of dummy jobs such that their new load is equal to the load of the machines with the second smallest load. We repeat this procedure until all dummy jobs in  $J_T$  are assigned. At the end, among all possibilities to assign the bags  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ , we choose the one which maximizes the minimum machine load after the distribution of  $J_T$ . We define  $\text{ALG}(m)$  as the load of the least loaded machine for this fixed candidate solution.

Among all guesses with the same set of bag sizes for bags in  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$ , the same value  $m_{\max}^{(K)}$  and the same values  $S$  and  $\bar{S}$ , we keep the guess which maximizes our proxy for the (partial) objective function,  $\sum_{m=1}^{m_{\max}^{(K)}} q_m \cdot \text{ALG}(m)$ .

#### 4.1.2 Analysis

Observing that  $|\mathcal{B}_K^*| \leq M \leq n$  and  $|\mathcal{B}_{K-1}^*| \leq M \leq n$  implies that we can enumerate all possible combinations in time  $O(n^2)$ . Since the relative length, i.e., the ratio of the left interval border to the right interval border, of  $I_K \cup I_{K-1}$  is bounded by  $(\frac{1}{\varepsilon})^6$ , there are at most  $\mathcal{O}_\varepsilon(1)$  possibilities for each size-estimate  $\ell(B)$ . By guessing the number of bags with a given size estimate, we can guess the size-estimates of all bags in  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$  in time  $n^{\mathcal{O}_\varepsilon(1)}$ . Further, each bag in  $\mathcal{B}_K^*$  can be assigned at most  $\mathcal{O}_\varepsilon(1)$  many jobs from  $J_K \cup J_{K-1}$  and, similarly, each bag in  $\mathcal{B}_{K-1}^*$  can be assigned at most  $\mathcal{O}_\varepsilon(1)$  many jobs from  $J_{K-1} \cup J_{K-2}$ . Hence, there is only a constant number of possible assignments for each bag, up to permutations of jobs with the same size. We formalize these observations in the next lemma.<sup>1</sup>

► **Lemma 10.** *In time  $n^{\mathcal{O}_\varepsilon(1)}$ , we can guess the size-estimate  $\ell(B)$  for each bag  $B \in \mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$  as well as the assignment of the jobs in  $J_K$  to the bags  $\mathcal{B}_K^*$ , of the jobs in  $J_{K-1}$  to the bags  $\mathcal{B}_K^* \cup \mathcal{B}_{K-1}^*$  and of a subset of jobs in  $J_{K-2}$  to the bags in  $\mathcal{B}_{K-1}^*$ , up to a permutation of bags.*

First, observe that for each bag  $B \in \mathcal{B}_K^*$ , the value  $\max\{[(1+\varepsilon)^{\ell(B)}] - p^+(B), 0\} \in \mathbb{N}_0$  since  $p_j \in \mathbb{N}$  for each  $j \in J$  by Lemma 9. Hence,  $S$  accurately captures the total volume missing to ensure that each  $B \in \mathcal{B}_K^*$  packs jobs with a total size of at least  $(1+\varepsilon)^{\ell(B)} \geq \frac{p(B)}{1+\varepsilon}$ . Using that each job in  $\bigcup_{k=1}^{K-2} J_k$  is very small compared to a bag in  $\mathcal{B}_K^*$ , we can argue that knowing  $S$  is actually sufficient to cover  $B \in \mathcal{B}_K^*$  with jobs of total size of at least  $\frac{p(B)}{(1+\varepsilon)^2}$ .

Similarly, for a bag in  $\mathcal{B}_{K-1}^*$ , each job in  $\bigcup_{k=1}^{K-3} J_k$  is very small compared to its size. Hence, we can again argue that knowing  $\bar{S} \in \mathbb{N}_0$  is sufficient to pack jobs of total size at least  $\frac{p(B)}{(1+\varepsilon)^2}$  into bag  $B \in \mathcal{B}_{K-1}^*$ .

Observing that no bag in  $\bigcup_{k=1}^{K-2} \mathcal{B}_k^*$  can pack a job from  $J_K \cup J_{K-1}$  by definition of their sizes, we conclude that  $T$  indeed represents the total volume of bags in  $\bigcup_{k=1}^{K-2} \mathcal{B}_k^*$ . In fact, we can show that for scenarios with  $m \leq m_{\max}^{(K)}$  machines *any* assignment of the remaining jobs in  $\bigcup_{k'=1}^{K-2} J_{k'}$  of total volume at most  $\frac{T}{1+\varepsilon}$  to at most  $M - |\mathcal{B}_K^*| - |\mathcal{B}_{K-1}^*|$  bags of size at most  $\varepsilon(\frac{1}{\varepsilon})^{3K}$  yields the same objective function value (up to a factor of  $1 + \mathcal{O}(\varepsilon)$ ). These observations are formalized in the next lemma where some jobs are set aside in bags  $B_S$  and  $B_{\bar{S}}$ , corresponding to the values  $S$  and  $\bar{S}$ .

Recall that we use  $\hat{X}$  to denote a possible guess for parameter  $X$  considered by our algorithm.

► **Lemma 11.** *Let the guessed quantities be as defined above. Let  $\mathcal{B}' \cup \{B_S, B_{\bar{S}}\}$  be a partition of the jobs  $\bigcup_{k'=1}^{K-2} J_{k'}$  into  $M - |\hat{\mathcal{B}}_K| - |\hat{\mathcal{B}}_{K-1}| + 2$  bags such that*

■ *for each bag  $B \in \mathcal{B}'$ ,  $p(B) \leq \varepsilon \cdot (\frac{1}{\varepsilon})^{3K}$ ,*

<sup>1</sup> For the initial guesses, one could give tighter bounds by observing that  $|\mathcal{B}_K^*| + |\mathcal{B}_{K-1}^*| = \mathcal{O}_\varepsilon(1)$ . However, we give polynomial bounds which are sufficient and of the same kind as the bounds we will use later in the DP.



- $p(B_S) \geq S$ ,
  - $p(B_{\bar{S}}) \geq \bar{S}$ , and
  - $p(\mathcal{B}') := \sum_{B \in \mathcal{B}'} p(B) \geq (1 + \varepsilon)^{-1} T$ .
- Suppose that  $B \in \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}$  has size in  $[(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$ . We can compute the vector  $(ALG(m))_{m=1}^{m_{\max}^{(K)}}$  in polynomial time and  $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m) \in [(1 + \varepsilon)^{-5} ALG(m), (1 + \varepsilon) ALG(m)]$  for each  $m \leq m_{\max}^{(K)}$ .

Note that the lemma does not state anything about the relationship of  $OPT(\mathcal{B}^*, m)$  and  $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m)$ ; it relates our proxy function  $ALG(m)$  and  $OPT(\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}, m)$ , the best possible assignment for  $\mathcal{B}' \cup \hat{\mathcal{B}}_K \cup \hat{\mathcal{B}}_{K-1}$ .

In the remaining problem it suffices to focus on scenarios in which we have  $m > m_{\max}^{(K)}$  machines and which hence satisfy  $OPT(\mathcal{B}^*, m) < (\frac{1}{\varepsilon})^{3K}$ . Note that for each bag  $B \in \mathcal{B}_K^*$  we have that  $p(B) \geq (\frac{1}{\varepsilon})^{3K}$ . Therefore, if we are given  $m > m_{\max}^{(K)}$  machines, it is optimal to assign each bag  $B \in \mathcal{B}_K^*$  to a separate machine without any further bags assigned to that machine. Hence, if  $m > m_{\max}^{(K)}$ , then the bags in  $\mathcal{B}_1^*, \dots, \mathcal{B}_{K-1}^*$  need to ensure only that the remaining  $m - |\mathcal{B}_K^*|$  machines get enough load. This insight and the above lemma allow us to decouple our decisions for scenarios with  $m \leq m_{\max}^{(K)}$  machines from scenarios with  $m > m_{\max}^{(K)}$  machines. This is the key idea for our DP.

## 4.2 Dynamic program

After our initial guesses above, it remains to

- pack the jobs in  $\bigcup_{k'=1}^{K-1} J_{k'}$  into the bags in  $\mathcal{B}_{K-1}^*$ ,
- compute the bag sizes in  $\mathcal{B}_1^*, \dots, \mathcal{B}_{K-2}^*$ , and
- select jobs from  $\bigcup_{k=1}^{K-2} J_k$  with total size at least  $S$  for filling  $\mathcal{B}_K^*$ .

For each  $m \geq m_{\min}^{(K-1)} := m_{\max}^{(K)} + 1$ , our goal is to obtain a value close enough to  $OPT(\mathcal{B}^*, m)$  so that, overall, we achieve a value of  $(1 - \mathcal{O}(\varepsilon))OPT$ .

To this end, we design a dynamic program (DP) that solves the remaining problem from above. Each DP cell corresponds to some subproblem. We show that for each possible guess of the initial quantities in Section 4.1 there is a DP cell corresponding to the remaining subproblem. In order to solve each subproblem, we guess similar quantities as in the previous section and reduce the resulting remaining problem to the subproblem of another DP cell.

### 4.2.1 Algorithm

Following the same idea as for the root subproblem, our dynamic program proceeds as follows: for each DP cell we first *guess* key quantities defining a partial solution as well as the transition to the next DP cell and then we *evaluate* this guessed partial solution.

**DP cell and its subproblem.** Formally, each DP cell  $\mathcal{C}$  is specified by

- $k \in \mathbb{N}$  with  $k < K$  specifying that we still need to define  $\mathcal{B}_1, \dots, \mathcal{B}_k$ ,
- $M_{k+1}, \dots, M_K \in \mathbb{N}$  counting the previously defined (large) bags  $\mathcal{B}_{k+1}, \dots, \mathcal{B}_K$ ,
- $M_k \in \mathbb{N}$  representing our decision  $|\mathcal{B}_k| = M_k$ ,
- $m_{\min}^{(k)} \in \mathbb{N}$  indicating the minimal number of machines we consider,
- $s_\ell \in \mathbb{N}$  for  $\ell \in L_k$  counting  $B \in \mathcal{B}_k$  with  $\ell(B) = \ell$ ,
- $a_\ell \in \mathbb{N}$  for  $\ell \in L_k$  counting the jobs  $j$  with  $p_j = \lceil (1 + \varepsilon)^\ell \rceil$  that are assigned to bags in  $\mathcal{B}_{k+1}$ ,

## 14:12 Scheduling on a Stochastic Number of Machines

- $S \in \mathbb{N}$ , defining the total size of jobs in  $\bigcup_{k'=1}^k J_{k'}$  that must not be assigned to bags  $\bigcup_{k'=1}^k \mathcal{B}_{k'}$  and that are neither assigned to bags in  $\mathcal{B}_{k+1}$  via the values  $a_\ell$ ; instead they will be assigned to  $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}$ . (Note that we will make sure that even though jobs from  $J_k$  might contribute to  $S$ , such jobs will not be used to cover bags in  $\mathcal{B}_{k+1}$ .)

The goal of the subproblem of  $\mathcal{C}$  is to pack a subset of the jobs  $\bigcup_{k'=1}^k J_{k'}$  into the bags  $\bigcup_{k'=1}^k \mathcal{B}_{k'}$  and define a size-estimate  $\ell(B)$  for  $B \in \bigcup_{k'=1}^k \mathcal{B}_{k'}$  such that

- $|\mathcal{B}_k| = M_k$ ,
- $p(B) \in I_{k'}$  for each  $k' \in [k]$  and each  $B \in \mathcal{B}_{k'}$ ,
- $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$  for each  $B \in \bigcup_{k'=1}^k \mathcal{B}_{k'}$ ,
- there are  $s_\ell$  bags  $B \in \mathcal{B}_k$  with  $\ell(B) = \ell$  for each  $\ell \in L_k$ ,
- each job  $j \in J_{k'}$  for  $k' \in [k]$  is either assigned to some bag in  $\mathcal{B}_{k'} \cup \dots \cup \mathcal{B}_k$  or not at all,
- there are  $a_\ell$  jobs  $j$  with  $p_j = \lceil (1 + \varepsilon)^\ell \rceil$  that are not assigned to any bag for each  $\ell \in L_k$ ,
- the jobs in  $\bigcup_{k'=1}^k J_{k'}$  not packed in any bag have total size at least  $S$ .

For each DP cell  $\mathcal{C}$ , we compute a solution and a corresponding objective function value which we denote by  $\text{profit}(\mathcal{C})$ . This objective function corresponds to the expected profit from scenarios in  $\{m_{\min}^{(k)}, \dots, M\}$  that we achieve with the solution stored in the DP cell and  $M_{k+1}, \dots, M_K$  “large” bags, i.e., bags  $B$  with  $p(B) \in \bigcup_{k'=k+1}^K I_{k'}$ .

**Guessing phase.** By definition of the DP cell,  $|\mathcal{B}_k^*| = M_k$ . For each  $\ell \in L_k$ , there are  $s_\ell$  many bags  $B \in \mathcal{B}_k^*$  with  $\ell(B) = \ell$  (and hence with  $p(B) \in [(1 + \varepsilon)^\ell, (1 + \varepsilon)^{\ell+1}]$ ). We start by guessing the assignment of the jobs  $J_{k-1} \cup J_k$  to the bags in  $\mathcal{B}_k^*$ . We only consider guesses satisfying the values  $a_\ell$  and  $S$  of our current DP cell, i.e., for every possible processing time in  $I_k$  enough jobs are left to be assigned to  $\mathcal{B}_{k+1}^*$  and enough total volume of jobs in  $\bigcup_{k'=1}^k J_{k'}$  is left to be assigned to bags in  $\bigcup_{k'=k+1}^K \mathcal{B}_{k'}$ . Finally, we guess  $m_{\max}^{(k)}$  which is the largest value  $m$  for which  $\text{OPT}(\mathcal{B}^*, m) \in I_k$ .

**Evaluation phase.** In order to calculate the proxy objective function value  $\text{profit}(\mathcal{C})$ , we need to combine  $\mathcal{C}$  with a cell  $\hat{\mathcal{C}}$  corresponding to a DP cell for the remaining problem. To this end, let us define the parameters of this cell  $\hat{\mathcal{C}}$ . Clearly, we only need to define  $\mathcal{B}_1, \dots, \mathcal{B}_{k-1}$ . Hence, the first parameter of  $\hat{\mathcal{C}}$  is  $k - 1$ . Further, the total number of previously defined bags is given by  $\hat{M}_{k, \dots, K} = M_k + M_{k+1}, \dots, M_K$ . As we do not ignore scenarios, we choose  $m_{\min}^{(k-1)} := m_{\max}^{(k)} + 1$ . Since we have already guessed the assignment of jobs in  $J_{k-1}$  to bags in  $\mathcal{B}_k^*$ , we can simply calculate the values  $\hat{a}_\ell$  for  $\ell \in L_{k-1}$  that indicate the number of jobs  $j$  with  $p_j = \lceil (1 + \varepsilon)^\ell \rceil$  to be assigned to bags in  $\mathcal{B}_k^*$ .

It remains to calculate the value  $\bar{S} \in \mathbb{N}$ , the total size of jobs in  $\bigcup_{k'=1}^{k-1} J_{k'}$  assigned as very small jobs to bags in  $\bigcup_{k'=k}^K \mathcal{B}_{k'}$ . To this end, we calculate the total size of jobs from  $\bigcup_{k'=1}^{k-2} J_{k'}$  that need to be packed in  $\mathcal{B}_k^*$ . For each  $B \in \mathcal{B}_k^*$ , let  $p^+(B)$  be the total size of the jobs from  $J_{k-1} \cup J_k$  that  $B$  already packs. We define  $S_k := \sum_{B \in \mathcal{B}_k^*} \max\{\lceil (1 + \varepsilon)^{\ell(B)} \rceil - p^+(B), 0\}$ . Denote by  $J_k(\mathcal{B}_k^* \cup \mathcal{B}_{k+1}^*)$  the set of jobs from  $J_k$  assigned to bags  $\mathcal{B}_k^*$  and  $\mathcal{B}_{k+1}^*$ . Then,  $\bar{S}$  is defined as  $\bar{S} := S - \sum_{j \in J_k \setminus J_k(\mathcal{B}_k^* \cup \mathcal{B}_{k+1}^*)} p_j + S_k$ , where  $S$  is defined by the current DP cell  $\mathcal{C}$ . (Note that  $\bar{S}$  does not contain jobs from  $J_{k-1}$  to be assigned to  $\mathcal{B}_k^*$  as they are accounted for by  $\hat{a}_\ell$ .)

Hence, the remaining problem corresponds to some DP cell  $\hat{\mathcal{C}}$  satisfying

$$\hat{\mathcal{C}} = \left( k - 1, M_{k+1}, \dots, M_K + M_k, \hat{M}_{k-1}, m_{\max}^{(k)} + 1, \hat{S} \geq \bar{S}, (\hat{s}_\ell)_{\ell \in L_{k-1}}, (\hat{a}_\ell)_{\ell \in L_{k-1}} \right), \quad (1)$$

where  $\hat{s}_\ell$  is a possible number of bags with size-estimate  $\ell \in L_{k-1}$ , i.e., with size  $(1 + \varepsilon)^\ell$ , the number of bags  $|\mathcal{B}_{k-1}^*|$  is given by  $\hat{M}_{k-1} = \sum_{\ell \in L_{k-1}} \hat{s}_\ell$ , and we require that  $\hat{S}$  is at least  $\bar{S}$ .

Given  $\text{profit}(\hat{\mathcal{C}})$  for some  $\hat{\mathcal{C}}$ , we can now calculate  $\text{profit}(\mathcal{C})$  as follows: For each value  $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$ , we compute an estimate  $\text{ALG}(m)$  of the objective value of an optimal bag-to-machines assignment of  $\bigcup_{k'=1}^k \hat{\mathcal{B}}_{k'}$  and  $M_{k+1}, \dots, K$  large bags to  $m$  machines. To this end, we use a variant of Lemma 11, which is explained in detail in the full version. Then, the profit of the candidate combination of  $\mathcal{C}$  with  $\hat{\mathcal{C}}$  is given by  $\sum_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}} q_m \text{ALG}(m) + \text{profit}(\hat{\mathcal{C}})$ . Among all these candidate combinations, we choose the one with the largest profit and set  $\text{profit}(\mathcal{C}) = \sum_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}} q_m \text{ALG}(m) + \text{profit}(\hat{\mathcal{C}})$ .

### 4.2.2 Analysis

Observe that there are at most  $\mathcal{O}_\varepsilon(1)$  many distinct processing times of jobs  $J_{k-1} \cup J_k$  and each bag  $B \in \mathcal{B}_k^*$  contains at most  $\mathcal{O}_\varepsilon(1)$  many jobs from each of these processing times because of the definition of  $\mathcal{B}_k^*$ ,  $J_{k-1}$ , and  $J_k$ .

We guess all possible assignments of jobs to a single bag of size at most  $(\frac{1}{\varepsilon})^{3k+3}$ ; typically such an assignment is called a *configuration*. There are at most  $\mathcal{O}_\varepsilon(1)$  such configurations. Then, for each configuration and each  $\ell$  with  $\lceil (1 + \varepsilon)^\ell \rceil \in I_k$ , we guess how often the configuration is assigned to a bag  $B$  with  $\ell(B) = \ell$ . Following this sketch, the next lemma proves that we can in fact guess the job-to-bag assignment in polynomial time.

► **Lemma 12.** *In time  $n^{\mathcal{O}_\varepsilon(1)}$  we can guess the assignment of jobs from  $J_{k-1}$  and  $J_k$  to the bags  $\mathcal{B}_k^*$  up to permuting jobs and bags.*

During the evaluation phase, we try all possible combinations of the current DP cell  $\mathcal{C}$  with  $\hat{\mathcal{C}}$  satisfying (1), i.e., DP cells corresponding to the remaining subproblems matching the parameters of  $\mathcal{C}$ . We now give a proof sketch of why our guesses combined  $\mathcal{C}$  indeed give a feasible solution to the subproblem for  $k$ . Let  $\hat{\mathcal{B}}_1, \dots, \hat{\mathcal{B}}_{k-1}$  be the bags given by the solution to  $\hat{\mathcal{C}}$  and  $\hat{\mathcal{B}}_k$  be the bags corresponding to our guess. (We do not change  $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$ .)

We need to assign jobs from  $\bigcup_{k'=1}^{k-1} J_{k'}$  to  $\hat{\mathcal{B}}_k$  satisfying

- (i) for every bag  $B \in \hat{\mathcal{B}}_k$ ,  $p(B) \geq (1 + \varepsilon)^{\ell(B)-1}$  and
- (ii) the total processing time of
  - all jobs in  $\bigcup_{k'=1}^{k-1} J_{k'}$  not assigned to bags in  $\bigcup_{k'=1}^k \hat{\mathcal{B}}_{k'}$  and
  - all jobs in  $J_k$  neither assigned to  $\hat{\mathcal{B}}_k$  nor reserved by the values  $a_\ell$  for the bags with size in  $I_{k+1}$
is at least  $S$ .

Each  $B \in \hat{\mathcal{B}}_k$  has already jobs from  $J_k$  and  $J_{k-1}$  of total size  $p^+(B)$  assigned to it. Let  $p^-(B) := \max\{\lceil (1 + \varepsilon)^{\ell(B)} \rceil - p^+(B), 0\}$  be the missing volume in  $B$  to cover  $B$  to the desired level of  $\lceil (1 + \varepsilon)^{\ell(B)} \rceil$ . If  $p^-(B) = 0$ , no additional job from  $\bigcup_{k'=1}^{k-2} J_{k'}$  needs to be assigned to  $B$ . Otherwise, we greedily add jobs from  $\bigcup_{k'=1}^{k-2} J_{k'}$  not packed in  $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$  until assigning the next job would exceed  $p^-(B)$ . Hence, the total size of jobs  $\bigcup_{k'=1}^{k-2} J_{k'}$  assigned to  $B$  by this routine is at least  $p^-(B) - (\frac{1}{\varepsilon})^{3k-6}$ . Thus,

$$p(B) \geq p^+(B) + p^-(B) - \left(\frac{1}{\varepsilon}\right)^{3k-6} = (1 + \varepsilon)^{\ell(B)} - \left(\frac{1}{\varepsilon}\right)^{3k-6} \geq (1 + \varepsilon)^{-1} (1 + \varepsilon)^{\ell(B)}$$

since by definition  $(1 + \varepsilon)^\ell \geq \frac{1}{\varepsilon} \cdot (\frac{1}{\varepsilon})^{3k-6}$  for all  $\ell \in L_k$ .

By choice of  $\hat{\mathcal{C}}$ , the total size  $\hat{S}$  of jobs in  $\bigcup_{k'=1}^{k-1} J_{k'}$  neither assigned to bags in  $\bigcup_{k'=1}^{k-1} \hat{\mathcal{B}}_{k'}$  nor reserved for  $\mathcal{B}_k$  via the values  $(\hat{a}_\ell)_{\ell \in L_{k-1}}$  is at least  $\bar{S}$ . With the definition of  $\bar{S}$ , we get

$$\hat{S} \geq \bar{S} = S - \sum_{j \in J_k \setminus J_k(\hat{\mathcal{B}}_k \cup \hat{\mathcal{B}}_{k+1})} p_j + S_k = S - \sum_{j \in J_k \setminus J_k(\hat{\mathcal{B}}_k \cup \hat{\mathcal{B}}_{k+1})} p_j + \sum_{B \in \hat{\mathcal{B}}_k} p^-(B).$$

We remark that the second term indeed corresponds to the contribution of jobs from  $J_k$  to filling bags with sizes in  $\bigcup_{k'=k+1}^K J_{k'}$  since such jobs cannot be packed into  $\bigcup_{k'=1}^{k-1} \hat{B}_k$  by definition of the corresponding sizes. Observe that the greedy procedure described above assigns jobs from  $\bigcup_{k'=1}^{k-2} J_{k'}$  with total volume *at most*  $p^-(B)$  to  $B \in \hat{B}_k$ . Hence, the combination of our guess  $\hat{B}_k$  (and the guessed partial assignment of  $J_{k-1} \cup J_k$  to  $\hat{B}_k$ ) and the solution for  $\hat{C}$  is indeed a feasible solution for  $\mathcal{C}$ .

Similar to the proof of Lemma 11, we show that for any candidate solution (consisting of a guess  $\hat{B}_k$ , a partial assignment of  $J_k \cup J_{k-1}$ , and any solution for  $\hat{C}$  as defined in (1)), we can calculate the values  $\text{ALG}(m)$  for  $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$  in polynomial time such that  $\text{ALG}(m)$  is within a factor  $(1 + \mathcal{O}(\varepsilon))$  of the optimal assignment given the same set of bags. This is formalized in the next lemma.

► **Lemma 13.** *Let  $\hat{B}_k, m_{\max}^{(k)}$  and the job-to-bag assignment of  $J_k \cup J_{k-1}$  to  $\hat{B}_k$  be guesses as defined. Further, let  $\hat{C}$  satisfy (1) and suppose that the bag sizes in  $\hat{B}_{k-1}$  are given by  $(\hat{s}_\ell)_{\ell \in L_{k-1}}$ . Let  $\mathcal{B}' \cup \{B_{\hat{s}}\}$  be a partition of the jobs  $\bigcup_{k'=1}^{k-2} J_{k'}$  into  $M - M_{k+1, \dots, K} - |\hat{B}_k| - |\hat{B}_{k-1}| + 1$  bags such that*

- for each bag  $B \in \mathcal{B}'$  we have that  $p(B) \leq \varepsilon \cdot \left(\frac{1}{\varepsilon}\right)^{3k}$ ,
- $p(B_{\hat{s}}) \geq \hat{S}$ ,
- $p(\mathcal{B}') := \sum_{B \in \mathcal{B}'} p(B) \geq (1 + \varepsilon)^{-1} T$ .

*Suppose that each bag  $B \in \hat{B}_k \cup \hat{B}_{k-1}$  satisfies  $p(B) \in [(1 + \varepsilon)^{\ell(B)}, (1 + \varepsilon)^{\ell(B)+1}]$  and let  $\hat{B}_L$  contain  $M_{k+1, \dots, K}$  many large bags of size at least  $\left(\frac{1}{\varepsilon}\right)^{3k+3}$ . There is a polynomial-time algorithm that either asserts that our guess is incorrect and cannot be combined with  $\hat{C}$  or that computes a vector  $(\text{ALG}(m))_{m=m_{\min}^{(k)}}^{m_{\max}^{(k)}}$  such that  $\text{OPT}(\mathcal{B}_L \cup \hat{B}_k \cup \hat{B}_{k-1} \cup \mathcal{B}', m) \in [(1 + \varepsilon)^{-5} \text{ALG}(m), (1 + \varepsilon) \text{ALG}(m)]$  holds for each  $m \in \{m_{\min}^{(k)}, \dots, m_{\max}^{(k)}\}$  for which  $\text{OPT}(\mathcal{B}_L \cup \hat{B}_k \cup \hat{B}_{k-1} \cup \mathcal{B}', m) \geq (1 + \varepsilon)^{-1} \left(\frac{1}{\varepsilon}\right)^{3k}$ .*

*Further, we can find the best  $\hat{C}$  that satisfies (1) and can be combined with our guess in polynomial time.*

To compute the final solution, we combine the correct initial guesses with the solution stored in the DP cell corresponding to the remaining subproblem. This yields a global solution to the original problem. In order to prove the correctness of our DP, we observe that for each  $k \in [K - 1]$  there is a *special* DP cell for which  $k$  is the first parameter and whose other parameters correspond to the optimal solution (e.g., the assignment of jobs in  $J_k$  to bags in  $\mathcal{B}_{k+1}^*$ ). We then prove by induction that, for each  $k \in [K - 1]$ , the solution stored in the corresponding special DP cell yields a profit that is similar to the optimal profit restricted to scenarios with  $m \in \{1, \dots, m_{\max}^{(k)}\}$  machines, using Lemma 13.

## 5 Conclusion

In this paper, we continue the recent line of research on scheduling with uncertainty in the machine environment [1, 5, 20] by considering a stochastic machine environment in which the number of identical parallel machines is only known in terms of a distribution and the actual number is revealed once the jobs are assigned to bags which cannot be split anymore. Interestingly, we present polynomial time approximation schemes for minimizing the makespan as well as maximizing the minimum machine load, which matches their respective deterministic counterparts from the perspective of approximation algorithms. We believe that our insights open up many interesting questions for future research such as extending the current model to the setting with uniformly related machines in which the uncertainty is modeled in terms of machine speeds as done in [5] from a robustness point-of-view or to the setting with different (job-based) objectives such as sum of weighted completion times.

---


**References**

---

- 1 Eric Balkanski, Tingting Ou, Clifford Stein, and Hao-Ting Wei. Scheduling with speed predictions. In *International Workshop on Approximation and Online Algorithms (WAOA)*, pages 74–89, 2023.
- 2 Anindya De, Sanjeev Khanna, Huan Li, and Hesam Nikpey. An efficient PTAS for stochastic load balancing with poisson jobs. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 37:1–37:18, 2020.
- 3 Florian Diedrich, Klaus Jansen, Ulrich M. Schwarz, and Denis Trystram. A survey on approximation algorithms for scheduling with machine unavailability. In *Algorithmics of Large and Complex Networks*, pages 50–64, 2009.
- 4 Franziska Eberle, Anupam Gupta, Nicole Megow, Benjamin Moseley, and Rudy Zhou. Configuration balancing for stochastic requests. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 127–141, 2023.
- 5 Franziska Eberle, Ruben Hoeksma, Nicole Megow, Lukas Nölke, Kevin Schewior, and Bertrand Simon. Speed-robust scheduling: sand, bricks, and rocks. *Math. Program.*, 197(2):1009–1048, 2023.
- 6 Leah Epstein and Rob van Stee. Maximizing the minimum load for selfish agents. *Theor. Comput. Sci.*, 411(1):44–57, 2010.
- 7 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.
- 8 Michael R. Garey and David S. Johnson. “Strong” NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978.
- 9 Anupam Gupta, Amit Kumar, Viswanath Nagarajan, and Xiangkun Shen. Stochastic load balancing on unrelated machines. *Math. Oper. Res.*, 46(1):115–133, 2021.
- 10 Anupam Gupta, Benjamin Moseley, and Rudy Zhou. Minimizing completion times for stochastic jobs via batched free times. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1905–1930, 2023.
- 11 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.
- 12 Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988. doi:10.1137/0217033.
- 13 Sharat Ibrahimpur and Chaitanya Swamy. Minimum-norm load balancing is (almost) as easy as minimizing makespan. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81:1–81:20, 2021.
- 14 Sungjin Im, Benjamin Moseley, and Kirk Pruhs. Stochastic scheduling of heavy-tailed jobs. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 474–486, 2015.
- 15 Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000.
- 16 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, 1976.
- 17 Jay Sethuraman. Stochastic scheduling. In *Encyclopedia of Algorithms*, pages 2110–2113. Springer, 2016.
- 18 Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Math. Oper. Res.*, 41(3):851–864, 2016.
- 19 Wolfgang Stadje. Selecting jobs for scheduling on a machine subject to failure. *Discret. Appl. Math.*, 63(3):257–265, 1995.
- 20 Clifford Stein and Mingxian Zhong. Scheduling when you do not know the number of machines. *ACM Trans. Algorithms*, 16(1):9:1–9:20, 2020.
- 21 Gerhard J. Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.*, 20(4):149–154, 1997.



# Distributional Online Weighted Paging with Limited Horizon

Yaron Fairstein  

Amazon.com, Haifa, Israel<sup>1</sup>

Joseph (Seffi) Naor 

Computer Science Department, Technion, Haifa, Israel

Tomer Tsachor  

Computer Science Department, Technion, Haifa, Israel

---

## Abstract

In this work we study the classic problem of online weighted paging with a probabilistic prediction model, in which we are given additional information about the input in the form of distributions over page requests, known as *distributional online paging* (DOP). This work continues a recent line of research on learning-augmented algorithms that incorporates machine-learning predictions in online algorithms, so as to go beyond traditional worst-case competitive analysis, thus circumventing known lower bounds for online paging. We first provide an efficient online algorithm that achieves a constant factor competitive ratio with respect to the best online algorithm (policy) for weighted DOP that follows from earlier work on the stochastic  $k$ -server problem.

Our main contribution concerns the question of whether distributional information over a limited horizon suffices for obtaining a constant competitive factor. To this end, we define in a natural way a new predictive model with limited horizon, which we call *Per-Request Stochastic Prediction* (PRSP). We show that we can obtain a constant factor competitive algorithm with respect to the optimal online algorithm for this model.

**2012 ACM Subject Classification** Theory of computation → Caching and paging algorithms; Theory of computation → Probabilistic computation; Theory of computation → Linear programming

**Keywords and phrases** Online algorithms, Caching, Stochastic analysis, Predictions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.15

**Category** APPROX

**Funding** *Joseph (Seffi) Naor*: Supported in part by Israel Science Foundation grant 2233/19 and United States – Israel Binational Science Foundation (BSF) grant 2033185.

## 1 Introduction

In the weighted paging problem there is a universe of  $n$  pages, where each page has a weight<sup>2</sup>, and there is a cache that can hold up to  $k$  pages. At each time step a page is requested, and if the requested page is already in the cache then no cost is incurred, otherwise, the page must be loaded into the cache, incurring a cost equal to its weight. The goal is to minimize the total cost incurred.

Paging is one of the earliest and most extensively studied problems in online computation and competitive analysis [38, 22, 40, 42, 36, 7, 6, 1, 4, 11, 10, 27, 28, 29], including works on non-standard caching models, e.g., elastic caches [25], caching with time windows [26], caching with dynamic weights [21], and caching with machine learning predictions [35]. In fact, online paging has become a focal point for many of the recent developments in competitive analysis, e.g., the online primal-dual method, projections, and mirror descent [17, 16, 15].

---

<sup>1</sup> Work done while at the Technion before joining Amazon.

<sup>2</sup> In the unweighted version of the problem, all weights are equal (unit weights).



In their seminal paper, Sleator and Tarjan [38] showed that any deterministic online algorithm is at least  $k$ -competitive and that the LRU policy (Least Recently Used) is exactly  $k$ -competitive for unweighted paging. The  $k$ -competitive bound was later generalized to weighted paging as well [19, 41]. When randomization is allowed, Fiat et al. [22] gave the elegant randomized marking algorithm for unweighted paging, which is  $\Theta(\log k)$ -competitive against an oblivious adversary. Bansal et al. [7] gave a  $\Theta(\log k)$ -competitive randomized algorithm for weighted paging based on the online primal-dual framework [17].

Algorithms with predictions, or learning-augmented algorithms, is an emerging field of research lying at the intersection of machine learning and foundations of algorithms (e.g. [5]). The goal is to use machine-generated predictions, that can be either deterministic or probabilistic, so as to go beyond traditional worst-case competitive analysis, and relax the overly pessimistic assumption of not having any prior knowledge of the future. This is in line with recent momentum in deploying machine learning techniques for various applications, e.g., search, business processes, and health.

The study of online paging with predictions has been a catalyst for the development of this new field. In an influential paper, Lykouris et al. [35] studied a simple predictor that provides for each requested page the next time step in which it is requested again, called PRP (Per-Request Prediction). Clearly, for unweighted paging, PRP suffices for implementing Belady's algorithm. Lykouris et al. [35] analyzed the robustness of PRP.

Computationally, weighted paging is a very different problem from unweighted paging, since it requires more global information about the request sequence to obtain (near) optimal algorithms. For example, Belady's local rule suffices to define an optimal offline algorithm in the unweighted case, while a minimum cost flow procedure is needed for computing an optimal solution in the weighted case. This is also manifested in the online setting, where PRP does not improve on the competitive factor in the weighted case [31]. Even with precise PRP, any deterministic online algorithm remains  $\Omega(k)$ -competitive, and any randomized algorithm is  $\Omega(\log k)$ -competitive.

### Distributional Online Paging

We focus on probabilistic prediction models for online weighted paging. Suppose that an online algorithm is given *in advance*, for each time step  $t \in \{1, 2, \dots, T\}$ , a distribution over page requests at  $t$ . Thus, the request at time  $t$  is drawn according to  $D_t$ . The given distributions are assumed to be *independent* between different time steps and the distributions are not necessarily identical. This model is known in the literature as *distributional online paging*, or DOP [34], and it can be viewed as the probabilistic counterpart of PRP. DOP is also a special case of the stochastic uber problem studied by [20]. (See more about this problem in the sequel.)

Define the cost of an online algorithm (or policy) for DOP to be the expected cost taken over all possible request sequences, where the probability of a request sequence is determined by the distributions  $D_1, \dots, D_T$ . An optimal online algorithm minimizes the expected cost and is defined by a Markov Decision Process. It can be computed by either a dynamic program or a linear program. Unfortunately, the state space of the dynamic program, or the size of the linear program, is of exponential size in  $k$ , rendering it computationally impractical. The computational hardness of finding an optimal algorithm for DOP is still open to the best of our knowledge<sup>3</sup>.

---

<sup>3</sup> It is stated as an open problem in [13]. For general metrics (i.e., the  $k$ -server problem), [20] gives a simple reduction to prove hardness: the uniform distribution over the points in the metric is given at all times; thus, a solution to the  $k$ -median problem on the metric defines the placement of the servers in an optimal online algorithm.



For unweighted DOP, the work of Lund et al. [34] is seminal. They show that *full* information about the distributions in each time step is actually not needed in order to get a near-optimal online algorithm. Specifically, for general distributions over page requests, if the probability that  $p$  is requested before  $q$  is available for any pair of pages  $p$  and  $q$ , then this information can be leveraged to get an efficient and simple 4-competitive algorithm with respect to the best online algorithm. However, these ideas do not seem to generalize to the weighted setting, due to the more global nature of weighted paging, as indicated before.

## 1.1 Our Results

We study the weighted DOP problem. Our goal is to provide an *efficient* online algorithm that achieves a constant competitive factor with respect to the best online algorithm (policy) for weighted DOP. Our starting point is a linear program for DOP which is based on the work of [20] for the stochastic  $k$ -server problem. In the case of the paging problem, this linear program specifies to which cache slot a page is loaded. The linear program provides a lower bound on the cost of any non-adaptive algorithm for DOP. However, [20] show that an optimal non-adaptive algorithm can cost at most thrice the cost of an optimal online algorithm for DOP. In Section 3 we provide the details for rounding the  $k$ -server linear program, yielding a constant competitive algorithm for the weighted DOP problem.<sup>4</sup> This is summarized in the following theorem.

► **Theorem 1.** *There exists an efficient algorithm for weighted DOP with  $O(1)$ -competitive ratio.*

The constant competitive factor obtained in Theorem 1 strongly utilizes information about the page distributions over the *entire* time horizon. However, such distributional information may not always be available. Thus, a natural question is whether distributional information over a limited horizon suffices for obtaining a constant competitive factor. This is the main focus of our paper.

In [31], a new deterministic predictive model for online weighted paging is suggested, due to the weakness of PRP in the weighted setting, as indicated earlier. This model, called SPRP, assumes that when a page  $p$  is requested, the full request sequence up to the next request for  $p$  is revealed. It turns out that the SPRP predictive model is strong enough to obtain a 2-competitive algorithm for online weighted paging in the adversarial setting [31].

Our first contribution is a novel limited horizon distributional model which we call the *Per-Request Stochastic Prediction* (PRSP) model. In this model, at any point of time, the known horizon of (future) distributions guarantees that for each page  $p$  in the cache, the sum of the probabilities of requesting  $p$  (in the known horizon) is at least one. Interestingly, this model captures the property needed from a limited horizon distributional model in order to design a near-optimal online algorithm.

Note that in a deterministic setting,  $D_t$  is equal to zero for all pages, except for one page, for which it is equal to one. Thus, when PRSP is restricted to a deterministic setting, it is equivalent to the SPRP model of [31].

We show that any algorithm for (full horizon) weighted DOP can be used in a black-box manner in the PRSP model, while increasing the competitive factor only by a constant. We thus obtain the next theorem.

<sup>4</sup> It is interesting to note that a natural linear programming formulation for the weighted DOP problem that provides a lower bound on the best online algorithm has a large integrality gap which depends on the maximum page weight. Thus, the work of [20] manages to circumvent this gap by utilizing a stronger LP.

► **Theorem 2.** *If there exists an  $\alpha$ -competitive algorithm for weighted DOP, then there exists an  $O(\alpha)$ -competitive algorithm for weighted DOP under the PRSP model. I.e., there exists an efficient  $O(1)$ -competitive algorithm for weighted DOP under the PRSP model.*

To prove Theorem 2, we design an algorithm called Split-and-Solve. The algorithm is very natural and it splits the time horizon into phases, solving each one separately. Using the properties of the PRSP model, the phases are chosen such that at the beginning of the phase the distributions are known for all times in the phase. Therefore, each phase can be regarded as a weighted DOP instance. As the analysis of Split-and-Solve does not make any assumptions regarding the solutions of the phases, any algorithm for weighted DOP can be employed in a black-box manner (e.g., Theorem 1). However, as each phase commences, the cache is reset to be the final cache state of the optimal offline solution of the realization of the request sequence of the previous phase.

The difficulty with analyzing the Split-and-Solve algorithm is in stitching together the performance of the different phases. Even though the policy employed in each phase by itself may be optimal, note that the initial cache states (in each phase) may be very different from the corresponding cache states of the optimal online algorithm (which is familiar with the full horizon). In particular, since our caching problem is weighted, the gap (in terms of weight) between cache states can be arbitrarily large, and may also further lead to poor performance within the phase. However, the crucial ingredient for bounding the performance of each phase is the property of the PRSP model that guarantees that for each page in the cache the sum of the probabilities in the known horizon adds up to at least 1. Thus, the performance of each phase can be related to the performance of the optimal online algorithm in the phase with a multiplicative constant factor (together with an additive term). Using a global analysis that considers all phases, the loss incurred by the sum of the additive terms can be charged to the cost of the optimal online algorithm, yielding Theorem 2.

## 1.2 Related Work

The  $k$ -server problem generalizes the paging problem to arbitrary metric spaces. (In paging the underlying metric is a weighted star.) A natural generalization of DOP is distributional  $k$ -server, where in every time step there is a given distribution over the possible request point. This problem was studied by [20], who also introduced the stochastic Uber problem, where each request is defined by two points in the metric. The server satisfying a request must travel to the start point of the request and then to its end point, incurring a cost equal to the total distance traveled. [20] gave a constant competitive factor algorithm for the case where the metric is a line. For general metrics, they gave an  $O(\log n)$ -competitive algorithm, where  $n$  is the number of points in the metric.

Work on distributional paging goes back more than fifty years. Franaszek and Wagner [24] compared FIFO and LRU in a model where every request is drawn from a fixed probability distribution over time. Shedler and Tung [37] suggested a Markov model for generating requests. This model and its extensions were analyzed in [32]. They also showed a gap of  $\Omega(\log k)$  between optimal online and offline algorithms in the case of a uniform distribution.

Besides stochastic models, several paging models assuming partial knowledge of future requests have been studied. For example, [8] studied the PRP model of Lykouris et al. [35] (mentioned earlier) in the weighted setting. A very sophisticated algorithm is given in [8] for this model whose competitive factor is at most a logarithm of the number of weight classes. Other examples for models with predictions are paging with locality of reference [12, 23, 30], paging with lookahead [3, 14, 39] and interleaved paging [9, 18, 33].

## 2 Preliminaries

### 2.1 Distributional Online Paging

In the weighted paging problem there is a universe of  $n$  pages, denoted by  $P = \{p_1, p_2, \dots, p_n\}$ , and a cache of size  $k$ . The initial cache state is  $C_0 \subseteq P$ . Each page  $p$  is associated with a weight  $w_p$ , the cost of loading or evicting page  $p$  to the cache. For a sequence of requests  $\sigma_1, \sigma_2, \dots, \sigma_T$ , an algorithm  $\mathcal{A}$  determines a series of cache states  $C_1, \dots, C_T$ , such that  $\forall t : \sigma_t \in C_t$ . The cost of serving the request sequence by  $\mathcal{A}$  is  $\sum_{t=1}^T \sum_{p \in C_t \Delta C_{t-1}} w_p$ , which is equal to the sum of the loading costs and the eviction costs. Note that the sum of the eviction costs and the loading costs can differ by at most an additive constant depending only on the initial and final cache contents. For simplicity, we assume the initial cache and final cache are identical, i.e.,  $C_0 = C_T$ . This means that the loading costs are equal to the eviction costs.

We focus on *distributional* prediction models in this paper. Suppose that we are given *in advance* at time  $t = 0$ , for each (future) time step  $t \in \{1, 2, \dots, T\}$ , a distribution over page requests at  $t$ . The given distributions are not necessarily identical, yet assumed to be *independent* between different time steps. More formally, for every  $t \in [T]$ , a probability distribution  $D_t$  is given from which request  $\sigma_t$  is drawn at time  $t$ . This model is called *distributional online paging*, or DOP.

► **Definition 3.** For an algorithm  $\mathcal{A}$ , let  $E(\mathcal{A})$  denote the expected cost of  $\mathcal{A}$  over all realizations of input sequences generated according to distributions  $D_1, \dots, D_T$ .

Denote by  $O$  the best online algorithm (in expectation) for DOP, i.e., it minimizes  $E(O)$ . We emphasize that algorithm  $O$  is only familiar with the distributions  $D_1, \dots, D_t$ , but does not have prior knowledge of the actual realization of the requests. Let  $E(O) = OPT$ .

### 2.2 Limited Horizon

So far we have assumed that all distributions  $D_1, \dots, D_T$  are given as input at the beginning. This is, of course, unreasonable in many cases as  $T$  may be very large. We aspire to bound the horizon that an online algorithm needs to “see”, i.e., at any time  $t$ , how many distributions into the future are available to the algorithm.

In [31] it was shown that the per-request-prediction (PRP) model and the lookahead model are not sufficient to circumvent the known lower bounds for online paging. In [31] a constant competitive factor for deterministic online weighted paging is achieved only through a combination of two models which they call SPRP. Specifically, upon arrival of a page, the time of its next request  $t$  is given, as in PRP, as well as the full sequence of page requests up to  $t$ .

Here, we propose an extension of the above model for our stochastic setting called Per-Request Stochastic Prediction (PRSP). The PRSP model requires that at any time  $t$  a sequence of future distributions is revealed such that sufficient information is revealed for each page in the current cache, as follows.

► **Definition 4.** Given a cache  $C \subseteq P$  and time  $t$ , let  $N(C, t)$  be the earliest time  $t'$  that satisfies:

$$\forall p \in C : \sum_{\tau=t}^{t'} D_\tau(p) \geq 1.$$

## 15:6 Distributional Online Weighted Paging with Limited Horizon

Note that in the deterministic setting,  $D_t$  is equal to zero for all pages, except for one page, for which it is equal to one. Assuming all pages in  $C$  are there because they were previously requested, Definition 4, restricted to a deterministic setting, is equivalent to the SPRP model of [31].

Using the definition of  $N(C, t)$  we can now give a formal definition of the PRSP model.

► **Definition 5.** *In the PRSP model at each time step  $t$ , where the cache is  $C_t$ , the sequence of distributions  $(D_t, \dots, D_{N(C_t, t)})$  is revealed to the algorithm.*

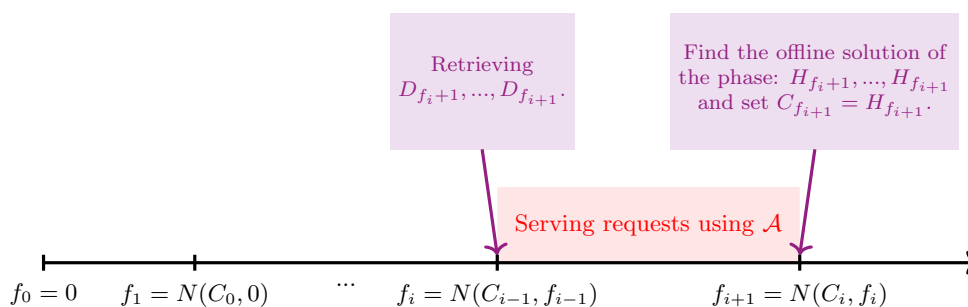
In a deterministic setting, at each time  $t$  it holds that there exists a single page  $p$  such that  $D_t(p) = 1$  and it is equal to zero for all other pages. Thus, in the PRSP model, when restricted to a deterministic setting, at any time  $t$ ,  $N(C_t, t)$  is equal to the latest time over the next arrivals of all pages in  $C_t$ . In reality, the distributions up to time  $N(C_t, t)$  were revealed earlier when the pages in  $C_t$  were requested and they were loaded to the cache. Thus, in the deterministic setting, we can interpret PRSP as SPRP since it reveals upon a request to a page the whole sequence of pages up to its next request.

### 3 Full Horizon

In this section we consider the weighted DOP problem when all the distributions are given in advance and provide a proof for Theorem 1. The proof essentially follows from the work of [20] on the stochastic  $k$ -server problem. Recall that weighted paging is the special case of  $k$ -server when the underlying metric is a weighted star. We show here that when applying the linear program for stochastic  $k$ -server to the special case of DOP, it can be rounded yielding a constant competitive algorithm. It is interesting to note that a natural linear programming formulation for the weighted DOP problem has a large gap compared to the best online algorithm, where the gap depends on the maximum page weight. Thus, the work of [20] manages to circumvent this gap by utilizing a stronger LP.

In [20], an algorithm  $\mathcal{A}$  for DOP is defined to be *non-adaptive* if it satisfies the following. First, algorithm  $\mathcal{A}$  pre-computes a sequence of cache configurations  $C_1, \dots, C_T$ ; then it serves the request sequence as follows. Upon arrival of request  $\sigma_t$  at time  $t$ : (i)  $\mathcal{A}$  changes the cache contents to configuration  $C_t$ ; (ii)  $\mathcal{A}$  replaces the lightest page in  $C_t$  with  $\sigma_t$ ; (iii)  $\mathcal{A}$  changes the cache contents back to the configuration that preceded the arrival of  $\sigma_t$ . The linear program for stochastic  $k$ -server suggested in [20] provides a lower bound on the cost of any non-adaptive algorithm. However, an online algorithm for the stochastic  $k$ -server problem does not necessarily imply a feasible solution for the latter linear program. It is shown in [20, Theorem 1.3] that an optimal non-adaptive online algorithm is a 3-approximation with respect to an optimal online algorithm for stochastic  $k$ -server. Thus, non-adaptive algorithms provide a useful tool for obtaining a competitive algorithm for stochastic  $k$ -server.

In what follows we describe the specialization of the linear program for the stochastic  $k$ -server problem [20] to DOP. Essentially, this means specifying to which cache slot is a page loaded. Thus, for each page  $p$ , there is a variable  $b_{t,p}$  that indicates the fraction of  $p$  that is not in the cache. For each page  $p$  and possible request  $q$ , variable  $x_{t,p,q}$  indicates whether  $q$  is served by replacing  $p$ .



■ **Figure 1** Algorithm *Split-and-Solve* splits the timeline into phases and separately solves each phase as DOP. As phase  $i$  begins at  $f_i$  the distributions till  $N(C_{f_i}, f_i)$  are revealed. The DOP black box  $\mathcal{A}$  serves the requests during the phase. As the phase terminates at  $N(C_{f_i}, f_i)$ , the optimal offline solution  $H_{f_{i+1}}, \dots, H_{f_{i+1}}$  is computed with the realization of the phase and  $H_{f_{i+1}}$  is loaded into  $C_{f_{i+1}}$ .

$$\begin{aligned}
 \min \quad & \sum_{p,q,t \geq 1} (w_p + w_q) \cdot D_t(q) \cdot x_{t,p,q} + \sum_{p,t \geq 1} w_p \cdot |b_{t,p} - b_{t-1,p}| \quad \text{s.t.} \\
 \forall t, q : \quad & \sum_{p \neq q} x_{t,p,q} \geq b_{t,q} \\
 \forall t, p, q : \quad & x_{t,p,q} \leq 1 - b_{t,p} \\
 \forall t : \quad & \sum_p b_{t,p} \geq n - k
 \end{aligned} \tag{31}$$

The cost function accounts for the weight of the pages that make two switches when serving a request. By doing so it forces the solution to be non-adaptive. Note that (31) gives us a lower bound only on the optimal non-adaptive algorithm. Thus, not every paging algorithm can be mapped into a solution for (31), only a non-adaptive one.

The details for rounding the LP solution so as to yield a 60-competitive algorithm are given in Appendix A.

## 4 The Split-and-Solve Algorithm

Section 3 provides us with a constant competitive factor algorithm, but requires that at time 0 all distributions  $D_1, \dots, D_T$  are known. In this section we present the Split-and-Solve algorithm that provides a constant-competitive factor for DOP in the PRSP model. The algorithm is very natural. It first splits the time horizon into phases: if a phase begins at time  $t$ , then it ends at time  $N(C_t, t)$ . As the horizon of the algorithm is at most  $\max\{N(C_{t'}, t') \mid t' \leq t\}$  for every time step  $t$ , the distributions  $D_{t+1}, \dots, D_{N(C_t, t)}$  can be retrieved by the properties of the PRSP model.

Each phase is solved independently using a (full horizon) DOP algorithm in a black-box manner. However, as each phase commences, the cache is reset to be the final cache state of the optimal offline solution of the realization of the request sequence of the previous phase. This cache state can be computed by running a min cost flow algorithm. This guarantees that the definition of phases is independent of the algorithm  $\mathcal{A}$ . The steps of the algorithm are depicted in Figure 1. We are now ready to give a formal definition of the Split-and-Solve algorithm.

---

**■ Algorithm 1 Split-and-Solve (SaS).**


---

**Input:** Instance  $\mathbf{I}$  of DOP and Algorithm  $\mathcal{A}$  for DOP.

- 1: Initialize  $i = 0, f_0 = 0, H_0 = C_0$ .
  - 2: **while**  $f_i < T$  **do**
  - 3:    Let  $f_{i+1} = N(C_{f_i}, f_i)$ .
  - 4:    Retrieve  $D_{f_{i+1}}, \dots, D_{f_{i+1}}$ .
  - 5:    Set  $C_{f_{i+1}}, \dots, C_{f_{i+1}}$  as the solution returned by  $\mathcal{A}$  for serving requests in time range  $[f_i, f_{i+1}]$  with initial cache  $H_{f_i}$ .
  - 6:    Let  $H_{f_{i+1}}, \dots, H_{f_{i+1}}$  be the optimal offline solution for time range  $[f_i, f_{i+1}]$  with initial cache  $C_{f_i}$ .
  - 7:    After serving  $\sigma_{f_{i+1}}$ , set  $C_{f_{i+1}} = H_{f_{i+1}}$  as the initial cache of the next phase.
  - 8:    Update  $i := i + 1$ .
- 

Before analyzing the algorithm, we need the following notation. We denote by  $F + 1$  the number of phases into which the time horizon is split. The optimal online algorithm is denoted by  $O_{on}$  and its caches are denoted by  $\{O_t\}_{t \in [T]}$ .

In the sequel we will show that given an  $\alpha$ -competitive algorithm for DOP, the Split-and-Solve (SaS) algorithm has an  $O(\alpha)$  competitive ratio, losing only an additional constant factor. To do so, we first bound in Lemma 7 the cost of our algorithm by the cost of the online algorithm when we also reset its cache at the beginning of each phase (i.e., similarly to Step 6, where at the beginning of each phase the cache is set to  $H_{f_i}$ ). Then, Lemma 8 bounds the cost of the online algorithm (with the cache reset) at phase  $i$  by the sum of three components:

1. The expected cost of  $O_{on}$  during the phase.
2. The total weight of  $H_{f_i} \setminus O_{f_i}$ .
3. The sum over pages in  $O_{f_i} \setminus H_{f_i}$  of the page weight times the probability it is requested in phase  $i$ .

Lemmas 9, 11 and 12 bound the last two components by a constant factor of the cost of  $O_{on}$ .

In the following definition we provide a notation for a partial solution.

► **Definition 6.** Let  $t_1, t_2 \in [T]$  such that  $t_1 < t_2$  and  $C \subseteq P$ . We denote the expected cost of an algorithm  $\mathcal{A}$  on the sub-range  $[t_1, t_2]$  with  $C_{t_1} = C$  as its initial cache by  $\mathcal{A}(C, t_1, t_2)$ .

From the definition of the optimal offline and online algorithms it is easy to see that for any  $t_1 < t_2$  and cache  $C$  it holds that  $O_{off}(C, t_1, t_2) \leq O_{on}(C, t_1, t_2)$ . The following lemma bounds the expected cost of phase  $i$  by  $2\alpha + 1$  times the expected cost of  $O$  during the phase when  $O$  begins the phase with the same cache.

► **Lemma 7.** Given an  $\alpha$ -competitive algorithm  $\mathcal{A}$  for DOP it holds that for every phase  $i$ :

$$\text{SaS}(H_{f_i}, f_i, f_{i+1}) \leq (2\alpha + 1) \cdot O_{on}(H_{f_i}, f_i, f_{i+1}).$$

**Proof.** At the beginning of each phase  $i$  we set the initial cache passed to algorithm  $\mathcal{A}$  in Step 5 to be the cache of the optimal offline solution  $H_{f_i}$ . For simplicity, we associate this cost with phase  $i - 1$  (note that for  $i = 0$  the cost is zero as  $H_0 = C_0$ ). Thus, in phase  $i$  we need to bound the cost of serving requests as well as the cost of loading cache  $H_{f_{i+1}}$  at the end of the phase.

The expected cost of serving requests in phase  $i$  is at most  $\alpha \cdot O_{on}(H_{f_i}, f_i, f_{i+1})$  due to the competitive ratio of  $\mathcal{A}$ . Next, loading  $H_{f_{i+1}}$  can be bounded by the cost of loading  $H_{f_i}$  and only then loading  $H_{f_{i+1}}$ . Loading  $H_{f_i}$  must cost less than the cost of SaS at this phase (as eviction and loading costs are symmetric). Afterwards, loading  $H_{f_{i+1}}$  is bounded by the cost of the optimal offline algorithm, but

$$O_{off}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(H_{f_i}, f_i, f_{i+1}).$$

Summarizing over the three cost components produces the desired bound.  $\blacktriangleleft$

The following lemma bounds  $O_{on}(H_{f_i}, f_i, f_{i+1})$  with the total cost of evicting  $H_{f_i} \setminus O_{f_i}$ , loading the requested pages from  $O_{f_i} \setminus H_{f_i}$  and then serving the requests as *on*.

► **Lemma 8.** *Let  $\gamma_{i,p}$  be the event that page  $p$  is requested in phase  $i$ , i.e.,  $\gamma_{i,p} = \mathbb{1}_p$  is requested in  $[f_i, f_{i+1}]$ . Then, it holds that:*

$$O_{on}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(O_{f_i}, f_i, f_{i+1}) + \sum_{p \in H_{f_i} \setminus O_{f_i}} w_p + \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right].$$

**Proof.** Consider the following online algorithm for serving phase  $i$ . First, evict the pages in  $H_{f_i} \setminus O_{f_i}$ , incurring a cost of  $\sum_{p \in H_{f_i} \setminus O_{f_i}} w_p$ . Next, run the optimal online algorithm. A feasible solution for the online algorithm would be to act as if  $O_{f_i} \setminus H_{f_i}$  are in the cache, incurring a cost of  $O_{on}(O_{f_i}, f_i, f_{i+1})$ . Nonetheless, it might be that a page  $p \in O_{f_i} \setminus H_{f_i}$  is requested, incurring an additional cost of  $w_p$ , though this only happens with probability  $\mathbb{E}[\gamma_{i,p}]$ . Summing over the three terms produces the desired bound.  $\blacktriangleleft$

After bounding the cost at each phase, we will now evaluate the cost of all phases, bounding the cost of splitting the time horizon. To do so we must bound  $\sum_i \sum_{p \in H_{f_i} \setminus O_{f_i}} w_p$  and  $\sum_i \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right]$ . The following lemma bounds the former term. It strongly uses the property of the PRSP model that for every page in cache the sum of the probabilities of requesting this page in the known horizon is at least 1.

► **Lemma 9.** *In each phase  $i \in [F]$  it holds that  $\sum_{p \in H_{f_i} \setminus O_{f_i}} w_p \leq \frac{e}{e-1} \cdot O_{on}(O_{f_i}, f_i, f_{i+1})$ .*

**Proof.** At the beginning of each phase we set  $f_{i+1} = N(H_{f_i}, f_i)$ . Thus from Definition 4 it holds that  $\sum_{t=f_i}^{f_{i+1}-1} D_t(p) \geq 1$  for each page  $p \in H_{f_i}$ . Using the inequality  $1 - x \leq e^{-x}$  for  $x \in [0, 1]$  we get that for  $p \in H_{f_i}$ :

$$\mathbb{E}[\gamma_{i,p}] = 1 - \prod_{t=f_i}^{f_{i+1}-1} (1 - D_t(p)) \geq 1 - \prod_{t=f_i}^{f_{i+1}-1} e^{-D_t(p)} = 1 - e^{-\sum_{t=f_i}^{f_{i+1}-1} D_t(p)} \geq 1 - e^{-1}. \quad (42)$$

Each page  $p \in H_{f_i} \setminus O_{f_i}$  must be loaded by  $O_{on}$  at phase  $i$  if it is requested. Thus from Equation (42) it follows that,

$$O_{on}(O_{f_i}, f_i, f_{i+1}) \geq \sum_{p \in H_{f_i} \setminus O_{f_i}} \mathbb{E}[\gamma_{i,p}] \cdot w_p \geq \sum_{p \in H_{f_i} \setminus O_{f_i}} (1 - e^{-1}) \cdot w_p.$$

Dividing by  $1 - e^{-1}$  completes the proof of the lemma.  $\blacktriangleleft$

The next corollary follows from lemmas 8 and 9.

► **Corollary 10.**

$$O_{on}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right].$$

## 15:10 Distributional Online Weighted Paging with Limited Horizon

Next, we bound  $\sum_i \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right]$ . To do so we need the following auxiliary lemma which bounds the expected eviction costs of  $O_{off}$ .

► **Lemma 11.** *Let  $Ev_{off}(H_{f_i}, f_i, f_{i+1})$  be the expected eviction costs of the optimal offline algorithm at phase  $i$  when initialized with cache  $H_{f_i}$ . It holds that,*

$$Ev_{off}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}).$$

**Proof.** Due to the optimality of the offline algorithm,  $O_{off}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(H_{f_i}, f_i, f_{i+1})$ . So from Corollary 10 it follows that,

$$O_{off}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]. \quad (43)$$

In addition, the offline optimal algorithm must load any page  $p \notin H_{f_i}$  in phase  $i$  if it is requested. Thus, the expected loading costs of the offline optimal algorithm are at least  $\mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]$ . By combining the bound on the loading costs and Equation (43) we get that,

$$\begin{aligned} Ev_{off}(H_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right] &\leq O_{off}(H_{f_i}, f_i, f_{i+1}) \\ &\leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[ \sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]. \end{aligned}$$

Subtracting the last term from both sides proves the lemma. ◀

► **Lemma 12.**

$$\mathbb{E} \left[ \sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}).$$

**Proof.** In Section 2.1 it is stated that we assume  $C_0 = C_T$ . If this was not the case, we can simply load  $C_T$  with only a constant additional cost. From this assumption it follows that the total eviction costs are equal to the total expected loading costs.

At each phase  $i$ ,  $O_{off}$  must load pages in  $O_{f_i} \setminus H_{f_i}$  if they are requested. Thus, in the event  $\gamma_{i,p}$  it will incur a loading cost of  $w_p$ . As the total expected eviction costs are equal to total expected loading costs we get that,

$$\mathbb{E} \left[ \sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} Ev_{off}(H_{f_i}, f_i, f_{i+1}).$$

Due to Lemma 11 it holds that

$$\mathbb{E} \left[ \sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}). \quad \blacktriangleleft$$

The following lemma combines the above results to produce a bound on the competitive ratio of SaS.



► **Lemma 13.** *Given an  $\alpha$ -competitive algorithm  $\mathcal{A}$  for DOP, the Split-and-Solve algorithm is  $(2\alpha + 1) \cdot \frac{4e-2}{e-1}$ -competitive.*

**Proof.** From Lemma 7 the cost of the Split-and-Solve algorithm is at most

$$\sum_{i \in [F]} (2\alpha + 1) \cdot O_{on}(H_{f_i}, f_i, f_{i+1}).$$

Combining with Corollary 10 and Lemma 12 which provides a bound on  $\sum_{i \in [F]} O_{on}(H_{f_i}, f_i, f_{i+1})$ , we can bound the competitive ratio of the Split-and-Save algorithm by  $(2\alpha + 1) \cdot \frac{4e-2}{e-1}$ . ◀

Theorem 2 follows immediately from Lemma 13.

---

## References

- 1 Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000. doi:10.1016/S0304-3975(98)00116-9.
- 2 Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An  $o(\log k)$ -competitive algorithm for generalized caching. *ACM Trans. Algorithms*, 15(1), November 2018. doi:10.1145/3280826.
- 3 Susanne Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997. doi:10.1007/PL00009158.
- 4 Susanne Albers, Sanjeev Arora, and Sanjeev Khanna. Page replacement for general caching problems. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 31–40. ACM/SIAM, 1999. URL: <http://dl.acm.org/citation.cfm?id=314500.314528>.
- 5 Algorithms with predictions (ALPS). <https://algorithms-with-predictions.github.io/>. Accessed: 2022-11-07.
- 6 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012. doi:10.1137/090779000.
- 7 Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):19, 2012.
- 8 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Scale-free allocation, amortized convexity, and myopic weighted paging. *CoRR*, abs/2011.09076, 2020. arXiv:2011.09076.
- 9 Rakesh D. Barve, Edward F. Grove, and Jeffrey Scott Vitter. Application-controlled paging for a shared cache. *SIAM Journal on Computing*, 29(4):1290–1303, 2000. doi:10.1137/S0097539797324278.
- 10 A. Blum, M. Furst, and A. Tomkins. What to do with your free time: algorithms for infrequent requests and randomized weighted caching, 1996.
- 11 Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 450–458. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814617.
- 12 A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995. doi:10.1006/jcss.1995.1021.
- 13 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
- 14 Dany Breslauer. On competitive on-line paging with lookahead. *Theor. Comput. Sci.*, 209(1-2):365–375, 1998. doi:10.1016/S0304-3975(98)00118-2.

- 15 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. 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 3–16. ACM, 2018. doi:10.1145/3188745.3188798.
- 16 Niv Buchbinder, Shahar Chen, and Joseph Naor. Competitive analysis via regularization. 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 436–444. SIAM, 2014. doi:10.1137/1.9781611973402.32.
- 17 Niv Buchbinder, Joseph Seffi Naor, et al. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.
- 18 Pei Cao, Edward W. Felten, and Kai Li. Application-Controlled file caching policies. In *USENIX Summer 1994 Technical Conference (USENIX Summer 1994 Technical Conference)*, Boston, MA, June 1994. USENIX Association. URL: <https://www.usenix.org/conference/usenix-summer-1994-technical-conference/application-controlled-file-caching-policies>.
- 19 Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, 1991. doi:10.1137/0220008.
- 20 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-server: How should uber work? In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPICs*, pages 126:1–126:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ICALP.2017.126.
- 21 Guy Even, Moti Medina, and Dror Rawitz. Online generalized caching with varying weights and costs. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 205–212. ACM, 2018. doi:10.1145/3210377.3210404.
- 22 Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991. doi:10.1016/0196-6774(91)90041-V.
- 23 Amos Fiat and Manor Mendel. Truly online paging with locality of reference. *CoRR*, abs/cs/0601127, 2006. arXiv:cs/0601127.
- 24 Peter A. Franaszek and T. J. Wagner. Some distribution-free aspects of paging algorithm performance. *J. ACM*, 21(1):31–39, 1974.
- 25 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Elastic caching. 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 143–156. SIAM, 2019. doi:10.1137/1.9781611975482.10.
- 26 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Caching with time windows. 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 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.
- 27 Sandy Irani. Competitive analysis of paging. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, pages 52–73. Springer, 1996. doi:10.1007/BFb0029564.
- 28 Sandy Irani. Page replacement with multi-size pages and applications to web caching. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 701–710. ACM, 1997. doi:10.1145/258533.258666.

- 29 Sandy Irani. Randomized weighted caching with two page weights. *Algorithmica*, 32(4):624–640, 2002. doi:10.1007/s00453-001-0095-6.
- 30 Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996. doi:10.1137/S0097539792236353.
- 31 Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. 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 69:1–69:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.69.
- 32 Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 208–217. IEEE Computer Society, 1992. doi:10.1109/SFCS.1992.267771.
- 33 Ravi Kumar, Manish Purohit, Zoya Svitkina, and Erik Vee. Interleaved caching with access graphs. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '20*, pages 1846–1858, USA, 2020. Society for Industrial and Applied Mathematics.
- 34 Carsten Lund, Steven J. Phillips, and Nick Reingold. Paging against a distribution and IP networking. *J. Comput. Syst. Sci.*, 58(1):222–232, 1999. doi:10.1006/jcss.1997.1498.
- 35 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/lykouris18a.html>.
- 36 Lyle A. McGeoch and Daniel Dominic Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991. doi:10.1007/BF01759073.
- 37 Gerald S. Shedler and C. Tung. Locality in page reference strings. *SIAM J. Comput.*, 1(3):218–241, 1972.
- 38 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.
- 39 Neal Young. *Competitive paging and dual-guided on-line weighted caching and matching algorithms*. Princeton University, 1991.
- 40 Neal E. Young. On-line caching as cache size varies. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*, pages 241–250. ACM/SIAM, 1991. URL: <http://dl.acm.org/citation.cfm?id=127787.127832>.
- 41 Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994. doi:10.1007/BF01189992.
- 42 Neal E. Young. On-line file caching. In *Symposium on Discrete algorithms*, pages 82–86, 1998.

## A Rounding the LP

We show how to discretize to multiples of  $1/k$  a solution to the LP. We use techniques from [2] and transform each  $x_{t,p}$  to a multiple of  $\frac{1}{8k}$ . First, We set  $b'_{t,p} := \min\{1, 2 \cdot b_{t,p}\}$ . Assume  $a_{t,p} \in [8k]$  denote  $\lceil \frac{8k \cdot b'_{t,p}}{8k} \rceil = \frac{a_{t,p}}{8k}$ . For every  $p \in P$ , we do the following iterative process for  $t = 1 \dots T$ :

1. If  $a_{t,p}$  is even, set  $y_{t,p} := \frac{a_{t,p}}{8k}$ .
2. Else, if  $y_{t-1,p} > \frac{a_{t,p}}{8k}$ , set  $y_{t,p} := \frac{a_{t,p}+1}{8k}$ .
3. Otherwise, set  $y_{t,p} := \frac{a_{t,p}-1}{8k}$ .

► **Lemma 14.** *The following statements hold:*

## 15:14 Distributional Online Weighted Paging with Limited Horizon

1.  $\forall t, p : \text{if } b_{t,p} = 0 \text{ then } y_{t,p} = 0.$
2.  $\forall t, p : \text{if } b_{t,p} = 1 \text{ then } y_{t,p} = 1.$
3.  $\forall t, p : y_{t,p} \leq 4 \cdot b_{t,p}.$
4.  $\forall p : \sum_{t \in [1, T]} |y_{t,p} - y_{t-1,p}| \leq 4 \cdot \sum_{t \in [1, T]} |b_{t,p} - b_{t-1,p}|.$
5.  $\forall t : \sum_{p \in P} y_{t,p} \geq n - k.$

**Proof.**

1.  $b_{t,p} = 0 \rightarrow a_{t,p} = 0 \rightarrow y_{t,p} = 0.$
2.  $b_{t,p} = 1 \rightarrow a_{t,p} = 8k \rightarrow y_{t,p} = 1.$
3. If  $b_{t,p} < \frac{1}{16k}$ , then  $a_{t,p} = 0$  implying  $y_{t,p} = 0$ . Otherwise,  $y_{t,p} \leq 2b_{t,p} + \frac{1}{8k} \leq 4b_{t,p}.$
4. For a page  $p \in P$ , we prove the claim by induction on  $T$ .

**Basis:** for  $t = 0$  the claim holds trivially.

**Inductive step:** Assume the claim holds for every  $\tau' < t$ . We assume w.l.o.g that  $a_{t,p} \leq a_{t-1,p}$ . We split to cases:

- a. If  $a_{t,p} = a_{t-1,p}$  then also  $y_{t,p} = y_{t-1,p}.$
- b. If  $a_{t,p} = a_{t-1,p} - 2$  then  $|b'_{t,p} - b'_{t-1,p}| \geq \frac{1}{8k}$ , in addition note that  $|y_{t,p} - y_{t-1,p}| = \frac{1}{4k}.$  Overall,  $|y_{t,p} - y_{t-1,p}| = \frac{1}{4k} \leq 2|b'_{t,p} - b'_{t-1,p}| \leq 4|b_{t,p} - b_{t-1,p}|.$
- c. If  $a_{t,p} \leq a_{t-1,p} - 3$  then  $|b'_{t,p} - b'_{t-1,p}| \geq \frac{1}{4k}$ , in addition  $|y_{t,p} - y_{t-1,p}| \leq |b'_{t,p} - b'_{t-1,p}| - \frac{1}{4k}.$  Overall,  $|y_{t,p} - y_{t-1,p}| \leq |b'_{t,p} - b'_{t-1,p}| - \frac{1}{4k} \leq 2|b'_{t,p} - b'_{t-1,p}| \leq 4|b_{t,p} - b_{t-1,p}|.$
- d. Else,  $a_{t,p} = a_{t-1,p} - 1$ . In case that  $a_{t,p}$  is odd we that  $y_{t,p} = y_{t-1,p}.$  Otherwise, let  $t'$  be the last time before  $t-1$  such that  $a_{t',p} \neq a_{t-1,p}.$  It holds that  $a_{t',p} \leq a_{t,p} - 2$  and we get, similar to Cases 4b and 4c,  $\sum_{i=t'+1}^t |y_{i,p} - y_{i-1,p}| = |y_{t,p} - y_{t',p}| \leq 2|b_{t,p} - b_{t',p}|.$

We show that we can find  $t' < t$  such that:

$\sum_{\tau \in [t', t]} |y_{\tau,p} - y_{\tau-1,p}| \leq 4 \cdot \sum_{\tau \in [t', t]} |b_{\tau,p} - b_{\tau-1,p}|.$  Finally, we apply the inductive assumption for  $t'.$

5. For time  $t$  we note  $A = \{p | b_{t,p} < 0.5\}.$  It holds for every  $p$  that if  $p \in A$  then  $y_{t,p} \geq 2 \cdot b_{t,p} - \frac{1}{8k}$ , else  $y_{t,p} = 1.$  Therefore, if  $|A| \leq k$  we are done. Else,  $\sum_{p \in P} y_{t,p} = \sum_{p \in A} y_{t,p} + \sum_{p \in P \setminus A} y_{t,p} \geq \sum_{p \in A} (2 \cdot b_{t,p} - \frac{1}{8k}) + |P| - |A| \geq 2(|A| - k) - \frac{|A|}{8k} + |P| - |A| = |P| + |A| - \frac{|A|}{8k} - 2k.$  Now, note that if  $k + 1 \leq |A| \leq 2k$ , then  $|A| \geq k + \frac{|A|}{8k}$  so  $|P| + |A| - \frac{|A|}{8k} - 2k \geq n - k.$  Else,  $|A| > 2k$  and  $|P| + |A| - \frac{|A|}{8k} - 2k \geq n - k. \quad \blacktriangleleft$

A similar process is required for discretizing the value of the  $x$  variables. For simplicity we assume there is an additional page  $z$  with weight 0 such that

$y_z = \max \left\{ 0, \sum_p y_{t,p} - (n + 1 - k) \right\}.$  For time  $t$  and pages  $p, q \neq z$  in  $P$ : if  $x_{t,p,q} \geq 0.5$  then  $v_{t,p,q} = 1 - y_{t,p}$ , else  $v_{t,p,q} = \min \{ 1 - y_{t,p}, x_{t,p,q} \}.$  For  $z$ ,  $v_{t,z,q} = \max \left\{ 0, y_{t,z} - \sum_p v_{t,p,q} \right\}.$

► **Lemma 15.** *The following statements hold:*

1.  $\forall t, p, q : v_{t,p,q} \leq 1 - y_{t,p}.$
2.  $\forall t, q : \sum_p v_{t,p,q} = y_{t,q}.$
3.  $\sum_{p,q,t \geq 1} (w_p + w_q) \cdot v_{t,p,q} \leq 2 \cdot \sum_{p,q,t \geq 1} (w_p + w_q) \cdot x_{t,p,q}.$

**Proof.**

1. For every page  $p \neq z$  the statement holds by definition. For  $z$ , since  $(1 - b_{t,q}) + \sum_p x_{t,p,q} \geq 1$ , we can view it as a rounding of a cache with (at least) one slot. The value of  $v_{t,p,z}$  is the empty space in the cache, which is at most the empty space in the fractional cache  $Y$ , i.e.  $1 - y_{t,z}.$
2. Holds immediately following definition of  $v_{t,z,q}.$

3. We can rewrite the sum:  $\sum_{p,q,t \geq 1} w_p \cdot v_{t,p,q} + \sum_{p,q,t \geq 1} w_q \cdot v_{t,p,q}$ . Since  $\sum_q v_{t,p,q} = y_{t,p}$  and  $y_{t,p} \leq 2b_{t,p}$ , implying  $\sum_{p,q,t \geq 1} w_p \cdot v_{t,p,q} \leq 2 \cdot \sum_{p,q,t \geq 1} w_p \cdot x_{t,p,q}$ . In addition, for every page  $q$ ,  $v_{t,p,q} \leq 2 \cdot x_{t,p,q}$ . ◀

We use Lemma 7.3 from [17] that provides a method for transforming distributions over pages into distributions over cache states. It is immediate from the proof of this lemma that if every distribution over the pages is a multiple of  $\frac{1}{L}$ , for some  $L \in \mathbb{N}$ , then the size of the distribution is polynomial in  $L$ ,  $n$  and  $T$ .

► **Definition 16.** For a page  $p$  and a cache  $C$ ,  $W(C, p) = 0$  if  $p \in C$ , otherwise  $W(C, p) = \min \{w_q | w \in C\}$ .

We state here a lemma that summarizes the desired construction and its properties.

► **Lemma 17.** Given a solution  $(B, X)$  to the LP, a collection of random integral cache states  $R(B, X) = \{R_1, \dots, R_T\}$  can be constructed in polynomial time (in  $n$ ,  $k$ , and  $T$ ) such that:

1.  $\forall t, p$ : if  $b_{t,p} = 0$ , then  $p \in R_t$ ; if  $b_{t,p} = 1$ , then  $p \notin R_t$ .
2.  $\forall t, p$ :  $\Pr[p \notin R_t] \leq 4 \cdot b_{t,p}$ .
3.  $\mathbb{E} \left[ \sum_{t \in [1, T], p \in R_t \Delta R_{t-1}} w_p \right] \leq 20 \cdot \sum_{t \in [1, T]} |b_{t,p} - b_{t-1,p}| \cdot w_p$ .
4.  $\forall t, p$ :  $\mathbb{E}[W(R(t), p)] \leq 8 \sum_{p,q,t \geq 1} w_p \cdot x_{t,p,q}$ .

We use the construction in the above lemma as a black box. When  $X$  is obvious from the context, we replace  $R(X)$  with  $R$ . Thus, we get the following algorithm:

■ **Algorithm 2** DOP Algorithm.

---

**Input:** Fractional solution  $B$  to LP (31).

- 1: Initialize: let  $R(B) = \{R_1, \dots, R_T\}$  (see Lemma 17).
  - 2: **for** time  $t$  and request  $\sigma_t$  **do**
  - 3:   Set  $C_t = R_t$ .
  - 4:   **if**  $\sigma_t \notin R_t$  **then**
  - 5:     Evicts the lightest page in  $R_t$  and loads  $\sigma_t$  instead.
  - 6:   Set  $C_t = R_t$ .
- 

► **Lemma 18.** Algorithm 2 is 60-competitive.

**Proof.** The expected cost of Step 3 is  $20 \cdot \sum_{p,t \geq 1} w_p \cdot |B_{t,p} - B_{t-1,p}|$ . In Step 5 the cost is 0 if  $\sigma_t \in R_t$  and  $w_{\sigma_t}$  otherwise. Therefore the expected cost of Step 5 at time  $t$  is  $\sum_{t,p} y_{t,p} \cdot w_p$ . Now let us assume for time  $t$  that  $\sigma_t \notin R_t$ . In this case, the cost of Step 6 is equal to  $\min \{w_q | q \in R_t\} = W(\sigma_t, C_t)$ . Therefore the expected cost of this step is  $\mathbb{E}[W(\sigma_t, R(t))]$ . From the construction of  $R_t$  in [17], this value is at most  $8 \cdot \sum_q w_q \cdot v_{t,\sigma_t,q}$ . In total, the expected cost of the algorithm is at most 20 times the optimal value of the linear program (31), hence at most 20 times the cost of the best non-adaptive algorithm. With [20, Theorem 1.3] we get that the expected cost is at most  $60 \cdot OPT$ . ◀


Note that all the cache configurations during the execution of the algorithm contain at most  $k$  pages. In addition, for every time  $t$ , the request  $\sigma_t$  is loaded in case it is not part of  $C_t$ .



# Weighted Matching in the Random-Order Streaming and Robust Communication Models

Diba Hashemi  

EPFL, Lausanne, Switzerland

Weronika Wrzos-Kaminska<sup>1</sup>  

EPFL, Lausanne, Switzerland

---

## Abstract

We study the maximum weight matching problem in the random-order semi-streaming model and in the robust communication model. Unlike many other sublinear models, in these two frameworks, there is a large gap between the guarantees of the best known algorithms for the unweighted and weighted versions of the problem.

In the random-order semi-streaming setting, the edges of an  $n$ -vertex graph arrive in a stream in a random order. The goal is to compute an approximate maximum weight matching with a single pass over the stream using  $O(n \text{ polylog } n)$  space. Our main result is a  $(2/3 - \epsilon)$ -approximation algorithm for maximum weight matching in random-order streams, using space  $O(n \log n \log R)$ , where  $R$  is the ratio between the heaviest and the lightest edge in the graph. Our result nearly matches the best known unweighted  $(2/3 + \epsilon_0)$ -approximation (where  $\epsilon_0 \sim 10^{-14}$  is a small constant) achieved by Assadi and Behnezhad [6], and significantly improves upon previous weighted results. Our techniques also extend to the related robust communication model, in which the edges of a graph are partitioned randomly between Alice and Bob. Alice sends a single message of size  $O(n \text{ polylog } n)$  to Bob, who must compute an approximate maximum weight matching. We achieve a  $(5/6 - \epsilon)$ -approximation using  $O(n \log n \log R)$  words of communication, matching the results of Azarmehr and Behnezhad [20] for unweighted graphs.

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Maximum Weight Matching, Streaming, Random-Order Streaming, Robust Communication Complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.16

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2408.15434>

**Acknowledgements** We thank Ola Svensson and Michael Kapralov for helpful discussions. We additionally thank Michael Kapralov for useful comments on the manuscript.

## 1 Introduction

The maximum matching problem is a fundamental problem in graph algorithms. In the unweighted version of the problem, we are interested in computing a maximum *cardinality* matching, i.e. to maximize the total number of edges in the matching. In the weighted version, we are interested in computing a maximum *weight* matching, i.e. to maximize the sum of the edge weights in the matching.

In this paper, we study matchings in the semi-streaming model. The semi-streaming model, originally introduced in [42], is motivated by the rise of massive graphs where the data is too large to be stored in memory, and has received extensive attention (see among others

---

<sup>1</sup> Corresponding author



© Diba Hashemi and Weronika Wrzos-Kaminska;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 16; pp. 16:1–16:26



Leibniz International Proceedings in Informatics

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

[69, 38, 48, 78, 58, 35, 73, 47, 59]). In this model, the edges of a graph arrive sequentially as a stream. The algorithm typically makes a single pass over the stream using space  $O(n \text{ polylog } n)$ , and must output an approximate maximum matching at the end of the stream. If the graph is unweighted, the greedy algorithm trivially gives a  $1/2$ -approximation, which is the best known for adversarially ordered streams. On the hardness side, it is known that a  $0.59$ -approximation is not possible [59] (see also [48, 58]). Closing the gap between these upper and lower bounds is one of the major open problems in the graph streaming literature. There has also been a long line of work on the weighted problem [42, 69, 38, 78, 35, 73, 47], culminating in a  $(1/2 - \epsilon)$ -approximation using space  $O(n)$  [73, 47].

Recently, there has been a wide interest in the random-order version of this problem, in which the arrival order of the edges is chosen uniformly at random. This problem has been extensively studied in the unweighted setting [64, 63, 5, 45, 41, 22, 6, 18]. Notably, Bernstein [22] gave a  $2/3$ -approximation, and Assadi and Behnezhad [6] improved it to  $(2/3 + \epsilon_0)$  for a small constant  $\epsilon_0 \sim 10^{-14}$ .

Progress on the weighted version of the problem lags behind. Gamlath et al. [45] broke the barrier of  $1/2$  in weighted graphs by obtaining a  $(1/2 + \delta)$ -approximation for a small constant  $\delta \sim 10^{-17}$ . More recently, Huang and Sellier [54] gave a  $\frac{1}{2-1/(2W)}$ -approximation under the assumption that the weights take integral values in  $[W]$ . This leaves a considerable gap between the best known results for the unweighted and weighted versions of the problem. In contrast, in other sublinear contexts, such as adversarially ordered streams or the dynamic graph setting, the weighted/unweighted gap has largely been closed [23]. The challenge of closing the gap in random-order streams remains an open problem, and has been highlighted explicitly in [22] and [23].

In this paper, we give a  $(\frac{2}{3} - \epsilon)$ -approximation algorithm for the weighted setting. Our result almost matches the best known  $(\frac{2}{3} + \epsilon_0)$ -guarantee for the unweighted setting, and improves significantly upon the previous results for the weighted setting.

► **Theorem 1.1.** *Given any constant  $\epsilon > 0$ , there exists a deterministic single-pass streaming algorithm that with high probability computes a  $(\frac{2}{3} - \epsilon)$ -approximate maximum weight matching if the edges arrive in a uniformly random order. The space usage of the algorithm is  $O(n \log n \log R)$ , where  $R$  is the ratio between the heaviest and the lightest edge weight in the graph.*

We also consider the two-player communication complexity model [77], and in particular the one-way communication complexity of matching, which was first studied in [48]. Here, the edge-set is partitioned between two parties Alice and Bob. Alice sends a single message to Bob, who must output an approximate maximum matching. Typically, we are interested in protocols with communication complexity  $O(n \text{ polylog } n)$ .

If the edges are partitioned adversarially between Alice and Bob, the right answer turns out to be  $2/3$ . A  $2/3$ -approximation can be achieved using  $O(n)$  communication for both bipartite unweighted [48], general unweighted [9] and general weighted [23] graphs. Going beyond a  $2/3$ -approximation requires  $n^{1+1/(\log \log n)} \gg n \text{ polylog } n$  communication even for unweighted bipartite graphs [48].

If instead the edges are partitioned randomly between the two parties, the answer is less clear. Recently, Azarmehr and Behnezhad [20] gave a  $5/6$ -approximation algorithm for unweighted graphs, improving upon a previous result of Assadi and Behnezhad [7]. To the best of our knowledge, prior to our work there were no results for weighted graphs (besides the  $2/3$ -approximation implied by adversarial protocols). We match the unweighted guarantees of Azarmehr and Behnezhad [20], thus closing weighted/unweighted gap in the robust communication complexity model.



► **Theorem 1.2.** *Given any constant  $\epsilon > 0$ , there exists a protocol that with high probability computes a  $(\frac{5}{6} - \epsilon)$ -approximate maximum weight matching in the two-party robust communication model using  $O(n \log n \log R)$  words of communication, where  $R$  is the ratio between the heaviest and the lightest edge weight in the graph.*

More generally, we match the results of Azarmehr and Behnezhad [20] for unweighted  $k$ -party robust communication, thus closing the unweighted/weighted gap also in this model.

► **Theorem 1.3.** *Given any  $k \geq 2$  and any constant  $\epsilon > 0$ , there exists a protocol that with high probability computes a  $(\frac{2}{3} + \frac{1}{3k} - \epsilon)$ -approximate maximum weight matching in the  $k$ -party one-way robust communication model using  $O(n \log n \log R)$  words of communication, where  $R$  is the ratio between the heaviest and the lightest edge weight in the graph.*

## 1.1 Related Work

The maximum matching problem is one of the most studied problems in the streaming setting, with numerous lines of work. This includes among others single-pass algorithms [42, 69, 38, 48, 78, 73, 47, 58, 35, 59, 8], multi-pass algorithms using 2 or 3 passes [64, 40, 56, 63, 65, 43, 3, 67, 66], and  $(1 - \epsilon)$ -approximation using a higher number of passes [69, 37, 1, 53, 2, 75, 45, 14, 10, 44, 18, 55, 4]. Garg et al. considered matching in a robust random-order streaming model with adversarial noise [46]. There are many results on dynamic streams, where edges can be deleted [33, 62, 12, 32, 11, 36, 17]. A different line of work considers estimating matching size, either in random-order streams [60, 28, 72, 61, 19] or in adversarially ordered streams [28, 70, 11, 34, 39, 71, 27, 13, 15]. Finally, there have also been several works on exact matching [42, 32, 53, 16, 30, 10].

## 2 Technical Overview

In this paper, we are interested in the random-order streaming model. The maximum cardinality matching problem has gained significant attention within this framework [64, 63, 45, 5, 41, 22, 6]. Bernstein [22] gave a  $2/3$ -approximation algorithm by adapting the “matching sparsifier” Edge-Degree Constrained Subgraph (EDCS) to the streaming context. Subsequent work by Assadi and Behnezhad [6] improved upon this, achieving a  $(2/3 + \epsilon_0)$ -approximation by simultaneously running Bernstein’s algorithm while identifying short augmenting paths. One of the motivations for studying the random-order setting, is that real-world data is rarely ordered adversarially. Rather, in most practical applications, it is reasonable to assume that the data is drawn from some distribution. However, assuming uniform randomness is often too strong of an assumption, since data correlations are prevalent in many real-world settings. This raises the question:

*How robust are random-order streaming algorithms to correlations in the arrival order?*

The robustness of random-order streaming algorithms to various types of adversarial distortions has already been studied previously, among others in the context of maximum matching and submodular maximization [46], rank selection [49, 50, 51], clustering problems [68] and component collection and counting [31]. In this paper, we focus on matchings. Our first contribution, is showing that existing algorithms for unweighted matching in random-order streams are in fact robust to correlations in the arrival order.

*Bernstein’s  $(\frac{2}{3} - \epsilon)$ -approximation algorithm is resilient to (limited) adversarial correlations in the arrival order.*

Surprisingly, this immediately gives a reduction from weighted matching in random-order streams.

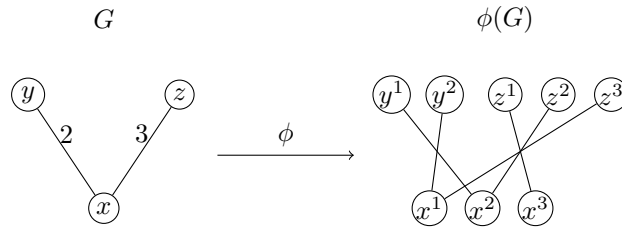
In *adversarially* ordered streams, Bernstein, Dudeja and Langley [23] gave a reduction from maximum weight matching to maximum cardinality matching. Progress in *random-order streams* has been comparatively limited. Gamlath et al. [45] achieved a  $(1/2 + \delta)$ -approximation, where  $\delta \sim 10^{-17}$  is a small constant. More recently, Huang and Sellier [54] gave a  $\frac{1}{2-1/(2W)}$ -approximation under the assumption that the weights take integral values in  $[W]$ , improving upon the result of Gamlath et al. [45] for small weights. They generalized the definition of EDCS to weighted graphs, which enabled them to adapt Bernstein's algorithm [22] to weighted graphs. However, their generalized notion of EDCS has weaker guarantees compared to the unweighted version, resulting in a significant loss in the approximation ratio.

Our second contribution is to nearly close the gap between weighted and unweighted maximum matching in random-order streams. We show that the reduction of Bernstein, Dudeja and Langley can be applied to random-order streaming algorithms which are resilient to specific correlations in the arrival order. This, together with the fact that Bernstein's algorithm [22] is robust to the appropriate correlations, gives a  $2/3$ -approximation algorithm for weighted bipartite graphs. We are also able to extend the guarantees to non-bipartite graphs.

## 2.1 Reduction in Adversarial Streams

First, we review the reduction of Bernstein, Dudeja and Langley [23] for adversarial streams. It is based on a technique called graph unfolding by Kao, Lam, Sung and Ting [57].

► **Definition 2.1** (Graph Unfolding [57]). *Let  $G = (V, E, w)$  be a graph with non-negative integral edge weights. The unfolded graph  $\phi(G)$  is an unweighted graph created as follows. For each vertex  $u \in V$ , let  $W_u = \max_{e \ni u} w_e$  be the maximum edge weight incident on  $u$ . There are  $W_u$  copies of  $u$  in  $\phi(G)$ , denoted by  $u^1, \dots, u^{W_u}$ . For each edge  $e = (u, v)$  in  $G$ , there are  $w_e$  edges  $\{(u^i, v^{w_e - i + 1})\}_{i \in [w_e]}$  in  $\phi(G)$ . See Figure 1 for an illustration.*



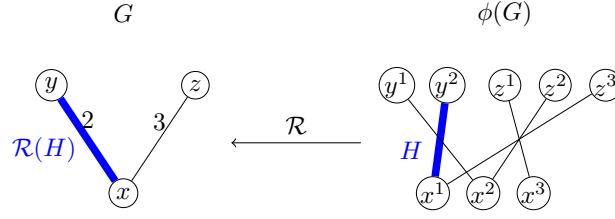
■ **Figure 1** An example of a weighted graph  $G$  and its unfolding  $\phi(G)$ .

One can also do a reverse operation of unfolding to bring a subgraph back to  $G$ .

► **Definition 2.2** (Refolding [23]). *Let  $G = (V, E)$  be a weighted graph and let  $H \subseteq \phi(G)$ . The refolded graph  $\mathcal{R}(H)$  has vertex set  $V$  and edge set  $E(\mathcal{R}(H)) := \{e = (u, v) \in G : (u^i, v^{w_e - i + 1}) \in H \text{ for some } i \in [w_e]\}$ . See Figure 2 for an illustration.*

Figure 1 illustrates the unfolding operation and Figure 2 illustrates the refolding operation. The key property of refolding is that it preserves the matching size in bipartite graphs.

► **Lemma 2.3** (Refolding preserves matching size in bipartite graphs [23]). *Let  $G$  be a weighted bipartite graph, and let  $H \subseteq \phi(G)$  be a subgraph of its unfolding. Then  $\mu_w(\mathcal{R}(H)) \geq \mu(H)$ .*



■ **Figure 2** An example of a subgraph  $H \subseteq \phi(G)$  and its refolding  $\mathcal{R}(H) \subseteq G$ . In this example,  $H = \{(u^1, v^2)\}$ . Then  $\mathcal{R}(H) = \{(u, v)\}$ .

This leads to a reduction from maximum weight bipartite matching to maximum cardinality bipartite matching in *adversarially ordered streams*: Upon arrival of each weighted edge  $e \in G$ , unfold  $e$  and pass the corresponding unweighted edges  $\phi(e)$  into an unweighted streaming algorithm. At the end of the stream we obtain an unweighted matching in  $\phi(G)$ , which we can refold to obtain a weighted matching in  $G$ .

In *random-order streams*, this reduction breaks for the following reason: For each weighted edge  $e \in G$ , the unweighted edges  $\phi(e)$  will necessarily arrive together. This introduces correlations in the arrival order of the edges, so the guarantees of random-order streaming algorithms do not apply. To address this, we consider a new streaming model, the *b-batch random-order stream model*, which is similar to the hidden-batch model introduced in [31]. This model allows us to capture the edge-correlations that arise from graph unfolding.

► **Definition 2.4** (*b-batch random-order stream model*). *In the b-batch random-order stream model the edge set of the input graph  $G = (V, E)$  is presented as follows: An adversary partitions the edge set  $E$  into batches  $\mathcal{B} = \{B_1, \dots, B_q\}$  with  $|B_i| \leq b$  for all  $i$ . The arrival order of the batches  $(B_{i_1}, \dots, B_{i_q})$  is then chosen uniformly at random among all the permutations of  $\mathcal{B}$ . The edges in each batch arrive simultaneously.*

Graph unfolding gives a reduction from weighted bipartite random-order streams to unweighted bipartite *b-batch random-order streams*. Each batch corresponds to one weighted edge, so given a weighted graph  $G$ , we can simply run a *b-batch random-order stream* algorithm on  $\phi(G)$  with batches  $\mathcal{B} = \{\phi(e) : e \in G\}$ .

## 2.2 Bernstein's Algorithm for Unweighted Random-Order Streams

We now review Bernstein's algorithm for unweighted random-order streams [22]. The algorithm proceeds in two Phases. Let  $\beta = O(\text{poly}(\epsilon^{-1}))$  be a parameter. Phase 1 constructs a subgraph  $H$  such that for all  $(u, v) \in H$ ,

$$\deg_H(u) + \deg_H(v) \leq \beta. \quad (1)$$

Given a subgraph  $H$ , we will say that an edge  $(u, v) \in G$  is *underfull* if  $\deg_H(u) + \deg_H(v) \leq \beta - 2$ , otherwise say that  $(u, v)$  is non-underfull.

The algorithm constructs  $H$  by adding underfull edges in a greedy manner, and then removing any edges that violate Equation 1. Phase 1 terminates when  $\approx \text{poly}(\epsilon) \frac{m}{n}$  non-underfull edges arrive in a row, and the algorithm then moves on to Phase 2. Bernstein [22] showed that it is only possible to make at most  $n\beta^2$  modifications to  $H$ . Since Phase 1 terminates when we see  $\approx \text{poly}(\epsilon) \frac{m}{n}$  edges in a row without modifying  $H$ , the Phase must terminate within the first  $\approx n\beta^2 \cdot \text{poly}(\epsilon) \frac{m}{n} \approx \epsilon m$  edges. This argument also holds in the *b-batch random-order stream model*.

## 16:6 Weighted Matching in Random-Order Streams

Then, in Phase 2, the algorithm simply collects all underfull edges into a separate set  $U$  (without modifying the graph  $H$ ). Let  $G_{late}$  denote the edges that arrive in Phase 2. Bernstein [22] proved the following structural result about  $H \cup U$ , which holds regardless of the assumptions on the arrival order:

$$\mu(H \cup U) \geq \left(\frac{2}{3} - \epsilon\right) \mu(G_{late}). \quad (2)$$

Since Phase 2 contains at least a  $(1 - \epsilon)$  fraction of the edges, and since the stream is uniformly at random, it follows from the Chernoff bound that  $\mu(G_{late}) \geq (1 - 2\epsilon)\mu(G)$ . Consequently, by Equation 2, it holds that

$$\mu(H \cup U) \geq \left(\frac{2}{3} - 3\epsilon\right) \mu(G).$$

For the space analysis, observe that  $H$  contains at most  $n\beta = O(n)$  edges. Let us now consider  $U$ . Recall that  $U$  is the set of all underfull edges that arrive after the termination of Phase 1, and that Phase 1 terminates when we see  $\approx \frac{m}{n}$  non-underfull edges in a row. So the only way for  $U$  to become too large, is if we draw  $\approx \frac{m}{n}$  non-underfull edges in a row when there are more than  $C \cdot n \log n$  underfull edges left in the stream, for some constant  $C$ . The probability of this event can be upper-bounded by

$$\left(1 - \frac{C \cdot n \log n}{m}\right)^{m/n} \leq n^{-C},$$

so with high probability, the algorithm stores at most  $O(n \log n)$  edges. Note that the space analysis breaks down in the  $b$ -batch random-order stream model, due to the correlated arrival orders.

### 2.3 Applying the Algorithm to Batch Arrivals

We now sketch why Bernstein's algorithm can be adapted to work under batch arrivals. Let  $b$  denote the upper-bound on the batch-size, and let  $q$  denote the total number of batches in the stream. Recall that in the reduction from weighted random-order streams,  $b$  corresponds to the maximum weight in the graph and  $q$  corresponds to the number of edges in the weighted graph. We will now describe how to obtain an algorithm with a polynomial space dependence on  $b$ . We will later discuss how to remove this dependence in the reduction from weighted random-order streams.

We will say that a batch is *underfull* if it contains at least one underfull edge. Otherwise, if it does not contain any underfull edges, say that it is non-underfull.

We terminate Phase 1 when  $\approx \text{poly}(\epsilon) \frac{q}{bn}$  non-underfull batches arrive in a row. This ensures that Phase 1 terminates within the first  $\approx n\beta^2 \cdot \text{poly}(\epsilon) \frac{q}{bn} \approx \frac{\epsilon}{b} q$  batches. Since each batch contains at most  $b$  edges, and since the arrival order of the batches is uniformly at random, it follows from Chernoff bounds that

$$\mu(G_{late}) \geq (1 - 2\epsilon)\mu(G).$$

Combining with Equation 2 we obtain

$$\mu(H \cup U) \geq \left(\frac{2}{3} - 3\epsilon\right) \mu(G).$$

The only way for the space usage to become too large, is if  $\approx \text{poly}(\epsilon) \frac{q}{b \cdot n}$  non-underfull batches arrive in a row when there are more than  $C \cdot n \log n \text{poly}(\frac{b}{\epsilon})$  underfull batches left in the stream, for some large constant  $C$ . The probability of this event can be upper-bounded by

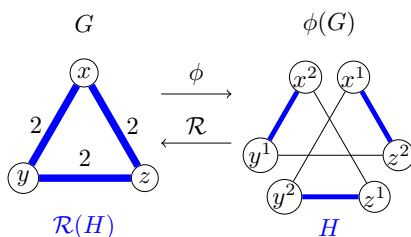
$$\left(1 - C \cdot \frac{n \log n}{q} \text{poly}\left(\frac{b}{\epsilon}\right)\right)^{\text{poly}(\epsilon/b)q/n} \leq n^{-C},$$

so with high probability, the algorithm stores at most  $O(n \log n \text{poly}(b))$  edges.

In our reduction, the parameter  $b$  corresponds to the maximum edge weight  $W$  in the graph. This means that we would incur a polynomial dependence on  $W$  in the space usage. However, a reduction due to Gupta and Peng [52] allows us to offset this space dependence. Gupta and Peng [52] devised a scheme for bucketing together edges according to their weight, which gives a reduction from general (possibly non-integral) weights, to integral bounded weights. Combining with this reduction, our algorithm uses space  $O(n \log n \log R)$ , where  $R$  is the ratio between the heaviest and the lightest edge in the graph, and it can handle any (possibly non-integral) edge weights. In particular, the space usage is  $O(n \text{polylog } n)$  as long as the weights are polynomial in  $n$ .

### 2.4 Non-bipartite graphs

In general, the reduction of Bernstein, Dudeja and Langley only holds for bipartite graphs. For non-bipartite graphs, it is no longer true that refolding preserves the matching size, since refolding a matching in  $\phi(G)$  can incur an additional  $2/3$  loss in the approximation ratio. Indeed, consider for example a weighted triangle with all edges of weight 2 (see Figure 3).



■ **Figure 3** Refolding does not in general preserve matching size in non-bipartite graphs. Consider for example the blue subgraph  $H = \{(x^1, z^2), (z^1, y^2), (y^1, x^2)\} \subseteq \phi(G)$  shown in the diagram. Then  $\mu(H) = 3$ , but  $\mu_w(\mathcal{R}(H)) = 2$ .

We prove that the subgraph  $H \cup U$  computed by Bernstein’s algorithm still satisfies  $\mu_w(\mathcal{R}(H \cup U)) \geq (2/3 - \epsilon)\mu_w(G)$ , even for non-bipartite graphs. This allows us to apply the unfolding reduction without any loss in the approximation ratio. We achieve this by reducing to the bipartite case: We show that for every weighted graph  $G$ , there exists a bipartite subgraph  $\tilde{G} \subseteq G$  such that  $\mu((H \cup U) \cap \phi(\tilde{G})) \geq (2/3 - \epsilon)\mu_w(G)$ . We can then apply Lemma 2.3 to the bipartite graph  $\tilde{G}$  to get the result.

In order to “bipartify” the graph, we use the following lemma from [23], which says that there exists a bipartite subgraph in which the degrees to  $H$  concentrate well (See Lemma 4.10 for the formal statement).

► **Lemma 2.5** (Informal version of Lemma 5.7 in [23]). *Let  $G$  be a weighted graph and let  $M^*$  be a maximum weight matching in  $G$ . Suppose that  $H \subseteq \phi(G)$  satisfies Equation 1. Then there exists a bipartite subgraph  $\tilde{G} \subseteq G$  such that  $\tilde{G}$  contains  $M^*$ , and, setting  $\tilde{H} := H \cap \tilde{G}$ , it holds that*

$$\deg_{\tilde{H}}(v) \approx \frac{\deg_H(v)}{2} \quad \forall v \in V.$$

Using this, we will show that  $(H \cup U) \cap \phi(\tilde{G})$  contains an EDCS, and therefore also contains a large matching.

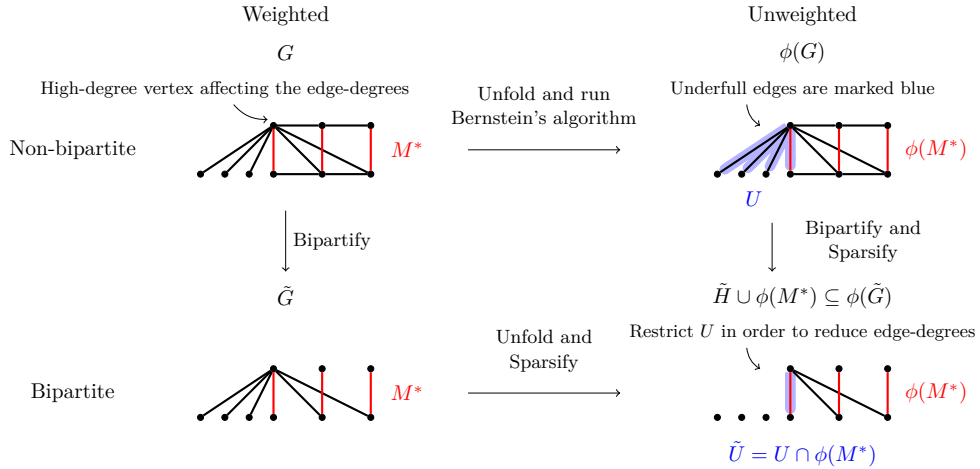
► **Definition 2.6** (EDCS [24]). *Let  $G = (V, E)$  be an unweighted graph, and  $H = (V, E_H)$  a subgraph of  $G$ . Given parameters  $\beta \geq 2$  and  $\lambda < 1$ , we say that  $H$  is a  $(\beta, \lambda)$ -EDCS of  $G$  if  $H$  satisfies the following properties:*

- (Property P1:) *For all edges  $(u, v) \in H$ , it holds that  $\deg_H(u) + \deg_H(v) \leq \beta$ .*
- (Property P2:) *For all edges  $(u, v) \in G \setminus H$ , it holds that  $\deg_H(u) + \deg_H(v) \geq \beta(1 - \lambda)$ .*

The crucial property of EDCS is that it contains a  $2/3$ -approximate maximum cardinality matching. This was first proved in [24] for bipartite graphs and in [25] for general graphs. See also Lemma 3.2 in [9] for a simpler proof with improved parameters.

► **Theorem 2.7** (EDCS contain a  $2/3$ -approximate matching [9]). *Let  $G$  be an unweighted graph and let  $\epsilon < 1/2$  be a parameter. Let  $\lambda, \beta$  be parameters with  $\lambda \leq \frac{\epsilon}{64}, \beta \geq 8\lambda^{-2} \log(1/\lambda)$ . Then, for any  $(\beta, \lambda)$ -EDCS  $H$  of  $G$ , we have that  $\mu(H) \geq (\frac{2}{3} - \epsilon)\mu(G)$ .*

Now consider the weighted input graph  $G$ . Fix a maximum weight matching  $M^*$  in  $G$  and let  $H$  be the graph computed by Phase 1 of Bernstein’s algorithm on input  $\phi(G)$ . Let  $\tilde{G} \subseteq G$  be the bipartite subgraph from Lemma 2.5. Ideally, we would like to show that  $(H \cup U) \cap \phi(\tilde{G})$  is an EDCS. However, this is not true in general, since the degrees to  $U$  can be arbitrarily large (consider for example the case when  $U$  is a star, see Figure 4 for an illustration), so  $\deg_{(H \cup U) \cap \phi(\tilde{G})}$  cannot be upper-bounded by a constant. Instead, we will sparsify  $U$ , so that its contribution to the degrees becomes insignificant. Let  $\tilde{H} = H \cap \phi(\tilde{G})$  and let  $\tilde{U} = U \cap \phi(M^*)$  (see Figure 4). This idea is similar to Bernstein’s original analysis [22], except that now we perform this sparsification in the unfolded and “bipartified” graph.



■ **Figure 4** Illustration of the reduction to the bipartite case. We show that  $\tilde{H} \cup \tilde{U}$  contains a matching of size at least  $(\frac{2}{3} - \epsilon)\mu_w(G)$ . Since  $\tilde{G}$  is bipartite, we can refold  $\tilde{H} \cup \tilde{U}$  without reducing the matching size.

Now  $\tilde{U}$  is a matching, so for all  $v \in V$ , we have  $\deg_{\tilde{H} \cup \tilde{U}}(v) \in \{\deg_{\tilde{H}}(v), \deg_{\tilde{H}}(v) + 1\}$ . So

$$\deg_{\tilde{H} \cup \tilde{U}}(v) \approx \deg_{\tilde{H}}(v) \approx \frac{1}{2} \deg_H(v).$$

In particular,

$$\forall (u, v) \in \tilde{H} \cup \tilde{U}, \quad \deg_{\tilde{H} \cup \tilde{U}}(u) + \deg_{\tilde{H} \cup \tilde{U}}(v) \approx \frac{1}{2} \deg_H(u) + \frac{1}{2} \deg_H(v) \leq \frac{\beta}{2},$$

and

$$\forall(u, v) \in \phi(M^*) \setminus (\tilde{H} \cup \tilde{U}), \quad \deg_{\tilde{H} \cup \tilde{U}}(u) + \deg_{\tilde{H} \cup \tilde{U}}(v) \approx \frac{1}{2} \deg_H(u) + \frac{1}{2} \deg_H(v) \geq \frac{\beta}{2} - 1.$$

Setting  $X = \tilde{H} \cup \tilde{U}$ ,  $\beta' \approx \frac{\beta}{2}$ , and  $\lambda'$  to be a sufficiently small constant, we can now apply Theorem 2.7 to the graph  $\tilde{H} \cup \phi(M^*)$ , and obtain

$$\mu(\tilde{H} \cup \tilde{U}) \geq (2/3 - \epsilon) \mu(\tilde{H} \cup \phi(M^*)) \geq (2/3 - \epsilon) \mu(\phi(M^*)).$$

Since  $\tilde{H} \cup \tilde{U} \subseteq \phi(\tilde{G})$  and since  $\tilde{G}$  is bipartite, we can apply Lemma 2.3 to get the required result

$$\mu_w(\mathcal{R}(H \cup U)) \geq \mu_w(\mathcal{R}(\tilde{H} \cup \tilde{U})) = \mu(\tilde{H} \cup \tilde{U}) \geq (2/3 - \epsilon) \mu(\phi(M^*)) = (2/3 - \epsilon) \mu_w(G).$$

In the rest of the paper, we will present the full analysis. In Section 4, we formally present the algorithm and analysis for random-order streams. In Section 5, we prove Theorem 1.2 and Theorem 1.3.

### 3 Notation and Preliminaries

Given a graph  $G = (V, E)$ , we will use  $n := |V|$  to denote the number of vertices and  $m := |E|$  to denote the number of edges in  $G$ . If  $G$  is weighted, then we will use  $w : E \rightarrow \mathbb{R}^+$  to denote the edge weights, and  $R := \max_{e \in E} w_e / \min_{e \in E} w_e$  to denote the ratio between the heaviest and the lightest edge in  $G$ . We use  $\mu(G)$  to denote the size of the maximum cardinality matching in  $G$ , and  $\mu_w(G)$  to denote the weight of the maximum weight matching in  $G$ .

Given  $\epsilon > 0$ , define  $\gamma_\epsilon := (4/\epsilon)^{\lceil 1/\epsilon \rceil}$ , a large constant which will be incurred in the space usage of our algorithms (instead of a dependence on the maximum weight of the graph). Note that for any fixed  $\epsilon$ , we have  $\gamma_\epsilon = O(1)$ .

#### 3.1 Models

**Random-order streams** In the random-order stream model, the weighted edges of the input graph arrive one-by-one in an order chosen uniformly at random from all possible orderings.

The algorithm makes a single pass over the stream and must output an approximate maximum weight matching at the end of the stream.

**Robust communication model** In the  $k$ -party one-way robust communication model, each weighted edge of the input graph is assigned independently and uniformly at random to one of the  $k$  parties. The  $i$ th party is supplied with its assigned edges and a message  $m_{i-1}$  from the  $(i-1)$ st party, and must send a message  $m_i$  to the  $(i+1)$ st party. The  $k$ th party must output a valid weighted matching of the input graph. The communication complexity of a protocol is defined to be  $\max_{1 \leq i \leq k} |m_i|$ , where  $|m_i|$  is the number of words in message  $m_i$ .

In the case of  $k = 2$ , we refer to the first party as Alice and to the second party as Bob.

#### 3.2 Graph Unfolding

In addition to the facts already stated in Section 2, we will need the following:

► **Theorem 3.1** (Unfolding preserves matching size in bipartite graphs [57]). *If  $G$  is a weighted bipartite graph, then  $\mu_w(G) = \mu(\phi(G))$ .*

► **Definition 3.2** (Refolding approximate [23]). *Let  $G$  be a weighted graph. A subgraph  $H \subseteq \phi(G)$  is  $\alpha$ -refolding-approximate if  $\mu_w(\mathcal{R}(H)) \geq \alpha \cdot \mu_w(G)$ .*

### 3.3 EDCS

We will use the following guarantee which holds for a relaxed notion of EDCS.

► **Definition 3.3** (Bounded edge-degree [22]). *We say that a graph  $H$  has bounded edge-degree  $\beta$  if for every edge  $(u, v) \in H$ , it holds that  $\deg_H(u) + \deg_H(v) \leq \beta$ .*

► **Definition 3.4** (Underfull edge [22]). *Let  $G$  be any unweighted graph, and let  $H$  be a subgraph of  $G$  with bounded edge-degree  $\beta$ . For any parameter  $\lambda < 1$ , we say that an edge  $(u, v) \in G \setminus H$  is  $(G, H, \beta, \lambda)$ -underfull if  $\deg_H(u) + \deg_H(v) < \beta(1 - \lambda)$ .*

► **Lemma 3.5** (Relaxed EDCS contain a 2/3-approximate matching [22]). *Let  $\epsilon < \frac{1}{2}$  be a parameter, and let  $\lambda, \beta$  be parameters with  $\lambda \leq \frac{\epsilon}{128}$ ,  $\beta \geq 16\lambda^{-2} \log(1/\lambda)$ . Consider any unweighted graph  $G$  and any subgraph  $H$  with bounded edge-degree  $\beta$ . Let  $U$  contain all edges in  $G \setminus H$  that are  $(G, H, \beta, \lambda)$ -underfull. Then  $\mu(H \cup U) \geq (2/3 - \epsilon)\mu(G)$ .*

### 3.4 Concentration Inequality

We will use the Chernoff bound for negatively associated random variables (see e.g. the primer in [76]).

► **Theorem 3.6.** *Let  $X_1, \dots, X_n$  be negatively associated random variables taking values in  $[0, 1]$ . Let  $X := \sum X_i$  and let  $\mu := \mathbb{E}[X]$ . Then, for any  $0 < \delta < 1$ , we have*

$$\Pr[X \leq \mu(1 - \delta)] \leq \exp\left(\frac{-\mu\delta^2}{2}\right).$$

## 4 2/3-Approximation in Random-Order Streams

In this section we prove Theorem 1.1. In Section 4.1, we formally describe the reduction from weighted random-order streams to unweighted  $b$ -batch random-order streams, and we prove its correctness. In Section 4.2, we show that Bernstein's 2/3-approximation algorithm [22] for random-order streams still works under batch-arrivals. Finally, in Section 4.3, we show that the obtained weighted random-order streaming algorithm still works for non-bipartite graphs, and we complete the proof of Theorem 1.1.

### 4.1 Reduction to Unweighted $b$ -batch Random-Order Streams

Gupta and Peng [52] gave a reduction which allows us to assume that the edge weights are integral and bounded above by a large constant. They originally proved the reduction for the dynamic graph model, however it also applies to the streaming and one-way communication models (See Theorem 6.1 and Theorem 6.2 in [23]).

► **Theorem 4.1** (Reduction to bounded integral weights [52, 23]). *If there is a random-order streaming algorithm  $\mathcal{A}$  to compute an  $\alpha$ -approximate maximum weight matching in graphs with edge weights in  $[W]$  and using space  $S(n, m, W, \alpha)$ , then there exists a random-order streaming algorithm  $\mathcal{A}'$  to compute a  $(1 - \epsilon)\alpha$ -approximate maximum weight matching with weights in graph with weights  $\mathbb{R}^+$  using space  $O(S(n, m, \gamma_\epsilon, \alpha) \log R)$ .*

*Similarly, if there is a one-way robust communication complexity protocol to compute an  $\alpha$ -approximate maximum weight matching for graphs with edge weights in  $[W]$  using  $C(n, m, W, \alpha)$  words of communication, then there exists a protocol to compute a  $(1 - \epsilon)\alpha$ -approximate maximum weight matching in graphs with weights in  $\mathbb{R}^+$  using  $O(C(n, m, \gamma_\epsilon, \alpha) \log R)$  words of communication.*



We would like to use the unfolding technique to reduce to the unweighted problem. As Bernstein, Dudeja and Langley [23] showed, in *adversarially ordered* streams, unfolding immediately gives a reduction for bipartite graphs: Whenever a weighted edge  $e \in G$  arrives, we can unfold it and pass the unweighted edges  $\phi(e)$  sequentially into an unweighted streaming algorithm while tracking the updates in the weighted stream. In *random-order* streams, there is a subtle issue with this approach. If the edges arrive uniformly at random in  $G$ , then the arrival order in  $\phi(G)$  will not be uniformly at random, but rather there will be batches of edges which necessarily arrive together. To overcome this issue, we consider the  $b$ -batch random-order stream model, restated below.

► **Definition 2.4** (*b*-batch random-order stream model). *In the  $b$ -batch random-order stream model the edge set of the input graph  $G = (V, E)$  is presented as follows: An adversary partitions the edge set  $E$  into batches  $\mathcal{B} = \{B_1, \dots, B_q\}$  with  $|B_i| \leq b$  for all  $i$ . The arrival order of the batches  $(B_{i_1}, \dots, B_{i_q})$  is then chosen uniformly at random among all the permutations of  $\mathcal{B}$ . The edges in each batch arrive simultaneously.*

Graph unfolding naturally gives a reduction from weighted random-order streams to unweighted  $b$ -batch random-order streams.

► **Theorem 4.2** (Reduction to the  $b$ -batch model). *If there exists an algorithm  $\mathcal{A}_B$  for the unweighted  $b$ -batch random-order stream model that computes an  $\alpha$ -approximate maximum cardinality matching in bipartite graphs using space  $S(n, m, b, \alpha)$ , then there exists an algorithm  $\mathcal{A}_w$  for weighted random-order streams (with weights in  $\mathbb{R}^+$ ) that computes a  $(1 - \epsilon)\alpha$ -approximate maximum weight matching in bipartite graphs using space  $O(S(n\gamma_\epsilon, m\gamma_\epsilon, \gamma_\epsilon, \alpha) \log R)$ .*

*Moreover, suppose that  $\mathcal{A}_B$  computes an  $\alpha$ -refolding approximate subgraph whenever the input graph is of the form  $\phi(G)$  for some weighted graph  $G$  with batches  $\mathcal{B} = \{\phi(e) : e \in G\}$ . Then the guarantees of  $\mathcal{A}_w$  also hold for non-bipartite graphs.*

**Proof.** Let  $\mathcal{A}_B$  be the unweighted  $b$ -batch random-order streaming algorithm using space  $S(n, m, b, \alpha)$ . By Theorem 4.1, it suffices to construct an algorithm  $\mathcal{A}_W$  that computes an  $\alpha$ -approximate maximum weight matching using space  $S(Wn, Wm, W, \alpha)$  when the edge weights are in  $[W]$ . We can obtain the required algorithm  $\mathcal{A}_W$  as follows:

Whenever an edge  $e$  arrives in the weighted stream, define a batch  $B_e := \phi(e)$  consisting of the unfolded edges of  $e$ . Feed the batch  $B_e$  as an update to the batch algorithm  $\mathcal{A}_B$ . In other words,  $\mathcal{A}_B$  is applied to the graph  $\phi(G)$  with batches  $\mathcal{B} = \{\phi(e) : e \in G\}$ . At the end of the stream,  $\mathcal{A}_B$  outputs an  $\alpha$ -approximate maximum cardinality matching  $M$  of  $\phi(G)$ . The algorithm  $\mathcal{A}_W$  outputs the maximum weight matching in  $\mathcal{R}(M)$  (which can easily be computed from  $M$ ). We have

$$\begin{aligned} \mu_w(\mathcal{R}(M)) &\geq \mu(M), && \text{by Lemma 2.3} \\ &\geq \alpha \cdot \mu(\phi(G)), && \text{by the assumption on } M \\ &= \alpha \cdot \mu_w(G), && \text{by Theorem 3.1} \end{aligned}$$

so  $\mathcal{A}_W$  outputs an  $\alpha$ -approximate maximum weight matching. Since the graph  $\phi(G)$  has at most  $Wn$  vertices and  $Wm$  edges, the space usage of  $\mathcal{A}_W$  is at most  $S(Wn, Wm, W, \alpha)$ , as required.

## 16:12 Weighted Matching in Random-Order Streams

For the “Moreover”-part, suppose that  $\mathcal{A}_B$  computes an  $\alpha$ -refolding approximate subgraph  $H$ . Define  $\mathcal{A}_W$  as before, except that now  $\mathcal{A}_W$  should output the maximum weight matching in  $\mathcal{R}(H)$ . Then

$$\mu_w(\mathcal{R}(H)) \geq \alpha \cdot \mu_w(G), \quad \text{by Definition 3.2,}$$

so the approximation ratio achieved by  $\mathcal{A}_W$  is still  $\alpha$ , even for non-bipartite graphs, as required.  $\blacktriangleleft$

► **Remark 4.3.** The argument can easily be adapted to the robust communication model. Consider a  $b$ -batch robust communication model, in which an adversary partitions the edges into batches of size at most  $b$ , and each batch gets assigned uniformly at random to each of the parties. Then any protocol for the  $b$ -batch robust communication model gives a protocol for the weighted robust communication model.

### 4.2 2/3-Approximation in $b$ -batch Random-Order Streams

In this section, we prove the following proposition.

► **Proposition 4.4.** *Given any unweighted graph  $G$  and any approximation parameter  $0 < \epsilon < 1$ , Bernstein’s algorithm (Algorithm 1) with high probability computes a  $(2/3 - \epsilon)$ -approximate maximum cardinality matching in the  $b$ -batch random-order stream model. The space complexity of the algorithm is  $O(nb^2 \log n \log b \text{ poly}(\epsilon^{-1}))$ , where  $b$  is the upper bound on batch-size.*

► **Definition 4.5 (Parameters).** *Let  $\epsilon < \frac{1}{2}$  be the final approximation parameter we are aiming for,  $\lambda = \frac{\epsilon}{512}$ ,  $\beta = 144\lambda^{-2} \log(2b/\lambda)$ ; note that  $\beta = O(\text{poly}(\epsilon^{-1}) \log b)$ . Set  $\alpha = \frac{\epsilon q}{b(n\beta^2 + 1)}$  and  $\gamma = 7 \log n \frac{q}{\alpha} = O(nb \log n \log b \text{ poly}(\epsilon^{-1}))$ .*

We now describe Bernstein’s algorithm [22] adapted to the  $b$ -batch model. The algorithm has two Phases. In Phase 1, it computes a subgraph  $H$  that is bounded edge-degree  $\beta$  (Definition 3.3). In Phase 2, it stores all the  $(G, H, \beta, \lambda)$ -underfull edges (Definition 3.4). That way, the algorithm computes a “relaxed” EDCS, which by Lemma 3.5 contains a  $(2/3 - \epsilon)$ -approximate maximum cardinality matching.

More concretely, the algorithm proceeds as follows: Initialize an empty subgraph  $H$  and start Phase 1. Phase 1 is split into epochs, each of which contains exactly  $\alpha$  batches. In each epoch, the algorithm processes the batches sequentially. For each edge  $(u, v)$  in the batch, if  $\deg_H(u) + \deg_H(v) < \beta(1 - \lambda)$ , then  $(u, v)$  is added to the subgraph  $H$  (note that the algorithm changes  $H$  over time, so  $\deg_H$  always refers to the degree of  $H$  at the time when the edge is examined). After adding an edge to  $H$ , the algorithm runs procedure REMOVEOVERFULLEDGES( $H$ ) to ensure that  $H$  is always bounded edge-degree  $\beta$ . In each epoch, the algorithm also has a boolean FOUNDUNDERFULL, which tracks whether at least one underfull edge arrived in the epoch. If FOUNDUNDERFULL is FALSE at the end of an epoch, then the algorithm terminates Phase 1 and moves on to Phase 2. Once Phase 1 terminates, the subgraph  $H$  becomes fixed and does not change anymore. Then, in Phase 2, the algorithm simply picks up all the underfull edges into a separate set  $U$ . For a formal description, see Algorithm 1. Note that the only difference between Algorithm 1 and Bernstein’s original algorithm (Algorithm 1 in [22]) is that the length of each epoch is now determined by the number of batches, rather than the number of edges.

► **Definition 4.6.** *Let  $\mathcal{B}_{\text{early}}$  denote the first  $\frac{\epsilon}{6}q$  batches in the stream and let  $\mathcal{B}_{\text{late}}$  denote the remaining batches. Let  $E_{\text{late}} := \{e \in E : e \in B \text{ for some } B \in \mathcal{B}_{\text{late}}\}$  be the set of edges that arrive as part of the late batches. More generally, let  $E_{>i}$  denote the the set of edges that arrive after the first  $i$  batches.*

■ **Algorithm 1** Bernstein’s Algorithm [22] adapted to the  $b$ -batch model.

---

```

1  $H \leftarrow \emptyset, U \leftarrow \emptyset$ 
2 Procedure PHASE 1
3   Do until termination
4     FOUNDUNDERFULL  $\leftarrow$  FALSE
5     for  $i = 1, \dots, \alpha$  do // Each epoch has  $\alpha$  batches
6       Let  $B_i$  denote the  $i^{\text{th}}$  batch in the epoch
7       for  $(u, v) \in B_i$  do
8         if  $\deg_H(u) + \deg_H(v) < \beta(1 - \lambda)$  then
9            $H \leftarrow H \cup \{(u, v)\}$ 
10          FOUNDUNDERFULL  $\leftarrow$  TRUE
11          REMOVEUNDERFULLEDGES( $H$ )
12        if FOUNDUNDERFULL = FALSE then
13          Go to Phase 2
14 Procedure REMOVEOVERFULLEDGES( $H$ )
15   while there exists  $(u, v) \in H$  such that  $\deg_H(u) + \deg_H(v) > \beta$  do
16     Remove  $(u, v)$  from  $H$ 
17 Procedure PHASE 2
18   foreach remaining edge  $(u, v)$  in the stream do
19     if  $\deg_H(u) + \deg_H(v) < \beta(1 - \lambda)$  then
20        $U \leftarrow U \cup \{(u, v)\}$ 
21   return the maximum matching in  $H \cup U$ 

```

---

First, we show that we don’t lose too many matching edges in the early part of the stream. To this end, we need to assume that the maximum cardinality matching is sufficiently large.

▷ **Claim 4.7.** We can assume that  $\mu(G) \geq 20b^2 \log n \epsilon^{-2}$ .

*Proof.* It is well known that every graph  $G$  satisfies  $m \leq 2n\mu(G)$ . Hence, if  $\mu(G) < 20b^2 \log n \epsilon^{-2}$ , then  $m = O(nb^2 \log n \epsilon^{-2})$ , so we can simply store all the edges. ◀

► **Lemma 4.8.** For  $\epsilon < b/2$ , it holds that  $\Pr[\mu(E_{\text{late}}) \geq (1 - 2\epsilon)\mu(G)] \geq 1 - n^{-5}$ .

*Proof.* Fix a maximum cardinality matching in  $M^*$  in  $G$ , and let  $B_1, \dots, B_k$  be the set of batches containing at least one matching edge from  $M^*$ . Each batch  $B_i$  contains at most  $b$  edges from  $M^*$ , so it suffices to show that at most  $\frac{2\epsilon\mu(G)}{b}$  of these batches arrive in the early part of the stream. For  $1 \leq i \leq k$ , let  $X_i$  be the indicator that  $B_i \in \mathcal{B}_{\text{late}}$ , and let  $X = \sum_{i=1}^k X_i$ . We will show that with high probability,  $X \geq k - \frac{2\epsilon\mu(G)}{b}$ . For  $1 \leq i \leq k$ , we have  $\mathbb{E}[X_i] = (1 - \frac{\epsilon}{b})$ , and so  $\mathbb{E}[X] = k(1 - \frac{\epsilon}{b})$ .

The  $X_i$ s are negatively associated, since these variables correspond to sampling  $(1 - \epsilon)q$  batches uniformly at random without replacement, which is known to be negatively associated (see the primer [76]). Applying Theorem 3.6 gives

$$\begin{aligned}
\Pr \left[ X \geq k - \frac{2\epsilon\mu(G)}{b} \right] &= 1 - \Pr \left[ X - \mathbb{E}[X] < - \left( \frac{2\epsilon\mu(G)}{b} - \frac{\epsilon k}{b} \right) \right] \\
&\geq 1 - \exp \left( \frac{-\epsilon^2(2\mu - k)^2}{4b^2k} \right) \\
&\geq 1 - n^{-5}.
\end{aligned}$$

The last inequality follows because  $\mu(G) \geq k$  and  $\mu(G) \geq 20b^2 \epsilon^{-2} \log n$ , by Claim 4.7. ◀

► **Lemma 4.9.** *Phase 1 satisfies the following properties:*

1. *Phase 1 terminates within the first  $\frac{\epsilon q}{b}$  batches of the stream.*
2. *Phase 1 constructs a subgraph  $H \subseteq G$  with bounded edge-degree  $\beta$ . As a corollary,  $H$  has at most  $O(n\beta)$  edges.*
3. *When Phase 1 terminates after processing some batch  $B_l$ , with probability at least  $1 - n^{-5}$ , the total number of  $(E_{>l}, H, \beta, \lambda)$ -underfull edges in  $E_{>l} \setminus H$  is at most  $b\gamma$ .*

The proof of Lemma 4.9 proceeds similarly to the proof of Lemma 4.1 in [22]. We will use the following result from the original analysis.

► **Lemma 4.10** ([22]). *Fix any parameter  $\beta > 2$ . Let  $H = (V, E_H)$  be a graph, with  $E_H$  initially empty. Say that an adversary adds and removes edges from  $H$  using an arbitrary sequence of two possible moves.*

- *(Deletion move) Remove an edge  $(u, v)$  from  $H$  for which  $\deg_H(u) + \deg_H(v) > \beta$ .*
- *(Insertion move) Add an edge  $(u, v)$  to  $H$  for some pair  $u, v \in V$  for which  $\deg_H(u) + \deg_H(v) < \beta - 1$ .*

*Then, after  $n\beta^2$  moves, no legal move remains.*

**Proof of Lemma 4.9.** *Property 1:* By Lemma 4.10, the algorithm can make at most  $n\beta^2$  changes to  $H$ . Since each epoch that does not terminate Phase 1 must make at least one change to  $H$ , there can be at most  $n\beta^2 + 1$  epochs in Phase 1. So overall, Phase 1 goes through at most  $\alpha(n\beta^2 + 1) = \frac{\epsilon q}{b}$  batches in Phase 1.

*Property 2:* Holds by construction of the algorithm, since the REMOVEOVERFULL procedure ensures that  $H$  is always bounded edge-degree  $\beta$ .

*Property 3:* Let  $l$  be the last batch processed in Phase 1. We will say that a batch  $B_j$  with  $j > l$  is *underfull* if it contains at least one  $(E_{>l}, H, \beta, \lambda)$ -underfull edge. We will show that with probability at least  $1 - n^{-5}$ , the number of underfull batches is at most  $\gamma$ . Since each underfull batch contains at most  $b$  underfull edges, this will give the result. The intuition is as follows: Phase 1 terminates only if there is an epoch without a single underfull batch. Since the stream is random, this means that there are relatively few underfull batches left in the stream. More formally, for each epoch  $0 \leq i \leq \frac{\epsilon q}{b}$ , say that a batch is *underfull* if it contains at least one  $(G, H_i, \beta, \lambda)$ -underfull edge, where  $H_i$  is the subgraph  $H$  constructed by the algorithm at the beginning of epoch  $i$ . Let  $\mathcal{E}_i$  be the event that no underfull batches appear in epoch  $i$ , and let  $\mathcal{F}_i$  be the event that there are more than  $\gamma$  underfull batches left in the stream when epoch  $i$  begins. Property 3 fails only if  $\mathcal{E}_i \wedge \mathcal{F}_i$  happens for some  $i$ , so we need to upper bound  $\Pr[\cup_{i=1}^{\epsilon q/b} \mathcal{E}_i \wedge \mathcal{F}_i]$ . Let  $\mathcal{B}_i^r$  denote the set of batches that have not yet appeared at the beginning of epoch  $i$  (r for remaining), let  $\mathcal{B}_i^e$  denote the set of batches that appear in epoch  $i$  (e for epoch) and let  $\mathcal{B}_i^u$  denote the set of underfull batches that remain in the stream (u for underfull). With these definitions, we can write  $\mathcal{E}_i \wedge \mathcal{F}_i$  as the event  $(\mathcal{B}_i^u \cap \mathcal{B}_i^e = \emptyset) \wedge (|\mathcal{B}_i^u| > \gamma)$ , since the event  $\mathcal{B}_i^u \cap \mathcal{B}_i^e = \emptyset$  ensures that the graph  $H$  does not change throughout the epoch. We have

$$\begin{aligned}
\Pr[\mathcal{E}_i \wedge \mathcal{F}_i] &= \Pr[(\mathcal{B}_i^u \cap \mathcal{B}_i^e = \emptyset) \wedge (|\mathcal{B}_i^u| > \gamma)] \\
&\leq \Pr[\mathcal{B}_i^u \cap \mathcal{B}_i^e = \emptyset \mid |\mathcal{B}_i^u| > \gamma] \\
&< \left(1 - \frac{\gamma}{q}\right)^\alpha \\
&= \left(1 - \frac{7 \log n}{\alpha}\right)^\alpha \\
&\leq n^{-7}.
\end{aligned}$$

Here the second inequality follows because  $\mathcal{B}_i^\epsilon$  is obtained by sampling  $\alpha$  batches from  $\mathcal{B}_i^r$  uniformly at random without replacement, and since  $|\mathcal{B}_i^u| > \gamma$  and  $|\mathcal{B}_i^r| \leq q$ . There are at most  $n^2$  epochs in total, so taking the union bound over all epochs gives the result.  $\blacktriangleleft$

Finally, we complete the proof of Proposition 4.4.

**Proof of Proposition 4.4.** Let us first show the approximation guarantee. By Part 2 of Proposition 4.9, Phase 1 computes a subgraph  $H$  which has bounded edge-degree  $\beta$ . Moreover, by Part 1 of Proposition 4.9, it holds that  $H \subseteq E_{late}$ . Phase 2 finds the set  $U$  of all  $(E_{late}, H, \beta, \lambda)$ -underfull edges. So by Lemma 3.5 applied to the graph  $E_{late}$ , the algorithm returns a matching of size at least

$$\begin{aligned} \mu(H \cup U) &\geq (2/3 - \epsilon)\mu(E_{late}) && \text{by Lemma 3.5} \\ &\geq (2/3 - \epsilon)(1 - 2\epsilon)\mu(G), && \text{by Lemma 4.8,} \end{aligned}$$

where the last inequality holds with probability at least  $1 - n^{-5}$ . Re-scaling  $\epsilon$  gives the approximation ratio.

For the space analysis: By Part 2 of Lemma 4.9, the space usage of Phase 1 is  $O(n\beta) = O(n \log b \text{poly}(\epsilon^{-1}))$ . By Part 3 of Lemma 4.9, with probability at least  $1 - n^{-5}$ , the space usage of Phase 2 is at most  $O(b\gamma) = O(nb^2 \log n \log b \text{poly}(\epsilon^{-1}))$ . So with probability at least  $1 - n^{-5}$ , the total space usage is at most  $O(nb^2 \log n \log b \text{poly}(\epsilon^{-1}))$ . By a union bound, the overall success probability of the algorithm is at least  $1 - 2n^{-5}$ .  $\blacktriangleleft$

### 4.3 Extension to Non-Bipartite Graphs

In this section, we show that the computed graph  $H \cup U$  is  $(2/3 - \epsilon)$ -refolding approximate. This, together with Proposition 4.4 and Theorem 4.2 will complete the proof of Theorem 1.1.

In [23], it was shown that EDCS are (almost)  $2/3$ -refolding approximate. However, since  $H \cup U$  is not actually an EDCS, but rather a relaxed version of an EDCS, this result cannot be applied directly. Instead, we need a more careful argument. We need the following lemma which was proved in [23].

► **Lemma 4.11** (Lemma 5.7 in [23]). *Let  $G$  be a weighted graph with weights in  $[W]$ . Let  $\delta \in (0, 1/2)$ , and let  $d \geq 36\delta^{-2} \log(W/\delta)$ . For any matching  $M$  in  $G$  and any subgraph  $H$  of  $\phi(G)$  with maximum degree at most  $d$ , there exists a bipartite subgraph  $\tilde{G} = \tilde{G}(M, H)$  of  $G$  such that, setting  $\tilde{H} := H \cap \phi(\tilde{G})$ , it holds that*

1.  $M \subseteq \tilde{G}$  and
2.  $|\deg_{\tilde{H}}(v) - \deg_H(v)/2| \leq \delta d$  for all vertices  $v \in V(H)$ .

► **Remark 4.12.** Bernstein, Dudeja and Langley [23] state the lemma for the special case when  $M$  is a maximum weight matching in  $G$ , however, without changing their argument, the same is true for any arbitrary matching  $M$ .

We now prove the main technical lemma, which shows that  $H \cup U$  is  $(2/3 - \epsilon)$ -refolding approximate.

► **Lemma 4.13.** *Let  $G$  be a (possibly non-bipartite) weighted graph with weights in  $[W]$ . Let  $\epsilon > 0$ ,  $\lambda \leq \frac{\epsilon}{512}$ ,  $\beta \geq \frac{144}{\lambda^2} \log(2W/\lambda)$ . Let  $G_S \subseteq G$  be any subgraph of  $G$ . Consider the unfolded graph  $\phi(G)$ . Let  $H$  be a subgraph of  $\phi(G)$  with bounded edge-degree  $\beta$ , and let  $U$  be the set of all edges in  $\phi(G_S) \setminus H$  that are  $(\phi(G_S), H, \beta, \lambda)$ -underfull. Then  $\mu_w(\mathcal{R}(H \cup U)) \geq (2/3 - \epsilon)\mu_w(G_S)$ .*

## 16:16 Weighted Matching in Random-Order Streams

**Proof.** Let  $\delta = \frac{\lambda}{2}$ . Fix a maximum-weight matching  $M^*$  of  $G_S$ , and let  $\tilde{G} = \tilde{G}(M^*, H)$  be the bipartite subgraph obtained from Lemma 4.11. Consider the subgraph  $\phi(\tilde{G}) \subseteq \phi(G)$ . Let  $\tilde{H} := H \cap \phi(\tilde{G})$  be the restriction of  $H$  to  $\phi(\tilde{G})$ , and let  $\tilde{U} := U \cap \phi(M^*)$  be the restriction of  $U$  to the matching  $\phi(M^*)$ . Note that  $\tilde{U}$  is a matching. By Lemma 4.11, we have  $\phi(M^*) \subseteq \phi(\tilde{G})$ , and therefore  $\tilde{H} \cup \tilde{U} \subseteq \phi(\tilde{G})$ . Therefore, we may now apply Lemma 2.3 to the bipartite graph  $\tilde{G}$  and the subgraph  $\tilde{H} \cup \tilde{U}$  of  $\phi(\tilde{G})$ .

$$\begin{aligned} \mu_w(\mathcal{R}(H \cup U)) &\geq \mu_w(\mathcal{R}(\tilde{H} \cup \tilde{U})), & \text{since } H \cup U \supseteq \tilde{H} \cup \tilde{U} \\ &\geq \mu(\tilde{H} \cup \tilde{U}), & \text{by Lemma 2.3.} \end{aligned} \quad (3)$$

Furthermore, recalling that  $M^*$  is a maximum weight matching in  $G_S$ , we have

$$\mu_w(G_S) = w(M^*) = |\phi(M^*)| \leq \mu(\tilde{H} \cup \phi(M^*)). \quad (4)$$

We will show that  $\mu(\tilde{H} \cup \tilde{U}) \geq (\frac{2}{3} - \epsilon)\mu(\tilde{H} \cup \phi(M^*))$ . To this end, we will show that  $\tilde{H} \cup \tilde{U}$  is an EDCS of  $\tilde{H} \cup \phi(M^*)$ .

▷ **Claim 4.14.**  $\tilde{H} \cup \tilde{U}$  is a  $(\beta', \lambda')$ -EDCS of  $\tilde{H} \cup \phi(M^*)$  for  $\beta' = \frac{\beta}{2} + \beta\lambda + 2$ ,  $\lambda' = 8\lambda$ .

*Proof.* Let us start by showing property P1 in Definition 2.6. First note that for all  $(u, v) \in \tilde{H} \cup \tilde{U}$ , it holds that  $\deg_H(u) + \deg_H(v) \leq \beta$ . Indeed, if  $(u, v) \in \tilde{H}$ , then  $(u, v) \in H$ , so  $\deg_H(u) + \deg_H(v) \leq \beta$  since  $H$  is bounded edge-degree  $\beta$ . If instead  $(u, v) \in \tilde{U}$ , then  $(u, v) \in U$ , so  $\deg_H(u) + \deg_H(v) \leq (1 - \lambda)\beta$ , since all elements of  $U$  are  $(\phi(G), H, \beta, \lambda)$ -underfull. Therefore, for all  $(u, v) \in \tilde{H} \cup \tilde{U}$ , it holds that

$$\begin{aligned} \deg_{\tilde{H} \cup \tilde{U}}(u) + \deg_{\tilde{H} \cup \tilde{U}}(v) &\leq \deg_{\tilde{H}}(u) + \deg_{\tilde{H}}(v) + \deg_{\tilde{U}}(u) + \deg_{\tilde{U}}(v) \\ &\leq \frac{\deg_H(u) + \deg_H(v)}{2} + 2\delta\beta + \deg_{\tilde{U}}(u) + \deg_{\tilde{U}}(v) \\ &\leq \frac{\beta}{2} + \beta\lambda + \deg_{\tilde{U}}(u) + \deg_{\tilde{U}}(v) \\ &\leq \frac{\beta}{2} + \beta\lambda + 2 \\ &= \beta'. \end{aligned}$$

The second inequality follows by Lemma 4.11, and the third inequality follows since  $\delta = \frac{\lambda}{2}$ .

We now show property P2 in Definition 2.6: If  $(u, v) \in (\tilde{H} \cup \phi(M^*)) \setminus (\tilde{H} \cup \tilde{U})$ , then  $(u, v) \in \phi(M^*) \setminus U$ , and in particular  $\deg_H(u) + \deg_H(v) > (1 - \lambda)\beta$  (by definition of  $U$ ). Thus,

$$\begin{aligned} \deg_{\tilde{H} \cup \tilde{U}}(u) + \deg_{\tilde{H} \cup \tilde{U}}(v) &\geq \deg_{\tilde{H}}(u) + \deg_{\tilde{H}}(v) \\ &\geq \frac{\deg_H(u) + \deg_H(v)}{2} - 2\delta\beta, & \text{by Lemma 4.11} \\ &\geq \frac{\beta(1 - \lambda)}{2} - \beta\lambda, & \text{since } \delta = \frac{\lambda}{2} \\ &\geq \left(\frac{\beta}{2} + \lambda\beta + 2\right)(1 - 8\lambda) \\ &= \beta'(1 - \lambda'). \end{aligned}$$

The last inequality follows from simple algebraic manipulations, using the fact that  $\lambda\beta \geq 2$ .

◁

By the choice of parameters, we have  $\lambda' \leq \frac{\epsilon}{64}$  and  $\beta' \geq 8\lambda'^{-2} \log(1/\lambda')$ , so Claim 4.14 together with Theorem 2.7 yields  $\mu(\tilde{H} \cup \tilde{U}) \geq (2/3 - \epsilon)\mu(\tilde{H} \cup \phi(M^*))$ . Combining everything, we get

$$\begin{aligned} \mu_w(\mathcal{R}(H \cup U)) &\geq \mu(\tilde{H} \cup \tilde{U}), && \text{by Equation 3} \\ &\geq (2/3 - \epsilon)\mu(\tilde{H} \cup \phi(M^*)) \\ &\geq (2/3 - \epsilon)\mu_w(G_S), && \text{by Equation 4.} \quad \blacktriangleleft \end{aligned}$$

Finally, we complete the proof of Theorem 1.1.

**Proof of Theorem 1.1.** We apply the reduction in Theorem 4.2 to Algorithm 1. By Proposition 4.4, Algorithm 1 computes a  $(2/3 - \epsilon)$ -approximate maximum cardinality matching using space  $O(n \log n \text{ poly}(b/\epsilon))$  in the  $b$ -batch random-order stream model. It remains to show that if the input graph is of the form  $\phi(G)$  for some weighted graph  $G$  with batches  $\mathcal{B} = \{\phi(e) : e \in G\}$ , then  $H \cup U$  is  $(2/3 - \epsilon)$ -refolding approximate. Let  $G_{late} \subseteq G$  denote the weighted edges corresponding to  $\mathcal{B}_{late}$ . An application of the Chernoff bound for negatively associated random variables (Theorem 3.6) shows that  $\Pr[\mu_w(G_{late}) \geq (1 - 2\epsilon)\mu_w(G)] \geq 1 - n^{-5}$  (the argument is similar to Lemma 4.8). Applying Lemma 4.13 to the graph  $G$  and the subgraph  $G_S := G_{late}$  yields

$$\begin{aligned} \mu_w(\mathcal{R}(H \cup U)) &\geq (2/3 - \epsilon)\mu_w(G_{late}), && \text{by Lemma 4.13} \\ &\geq (1 - 2\epsilon)(2/3 - \epsilon)\mu_w(G) \\ &\geq (2/3 - 3\epsilon)\mu_w(G), \end{aligned}$$

as required. Re-scaling  $\epsilon$  and applying Theorem 4.2 yields the result.  $\blacktriangleleft$

## 5 5/6-Approximation in the Robust Communication Model

In this section, we prove Theorem 1.2 and Theorem 1.3. By applying the results from the previous section, we can generalize the protocol of Azarmehr and Behnezhad [20] to the weighted case. By the reduction in Theorem 4.1, we can assume that the edge weights take integral values in  $[W]$ , for a large constant  $W$ . We will now describe the protocol for the two-party model.

Let  $\epsilon > 0$  be the final parameter we are aiming for, and let

$$\lambda = \frac{\epsilon}{2048}, \beta = 144\lambda^{-4} \log(2W/\lambda).$$

Let  $E_A$  denote the set of edges assigned to Alice and  $E_B$  the set of edges assigned to Bob. Alice simulates a random-order stream. She unfolds the edges and runs Algorithm 1 on the corresponding unweighted  $W$ -batch random-order stream. That way, she obtains a set  $H \subseteq \phi(E_A)$  with bounded edge degree  $\beta$  and a set  $U_A \subseteq \phi(E_A)$  consisting of all  $(\phi(E_A \setminus E_{early}), H, \beta, \lambda)$ -underfull edges, where  $E_{early} \subseteq E_A$  denotes the first  $\frac{\epsilon}{W}m$  weighted edges in her simulated stream. She communicates  $\mathcal{R}(H \cup U_A)$  to Bob. Bob outputs the maximum weight matching in  $\mathcal{R}(H \cup U_A) \cup E_B$ . See Algorithm 2 for a formal description.

The protocol for  $k$  parties is similar, only that now all of the first  $k - 1$  parties should simulate a random-order stream (we describe the protocol more formally in the proof of Theorem 1.3).

Assume that each edge is assigned to Bob with probability  $p \leq \frac{1}{2}$  (this will make the analysis applicable to the  $k$ -party setting). Let  $U_B$  be the set of all  $(\phi(E_B), H, \beta, \lambda)$ -underfull edges, i.e. the set of underfull edges assigned to Bob. Let  $U := U_A \cup U_B$  denote the set of all

## 16:18 Weighted Matching in Random-Order Streams

■ **Algorithm 2** Robust Communication Protocol for Weighted Graphs.

- 
1. Alice simulates a random-order stream by ordering the edges in  $E_A$  uniformly at random.
  2. Alice obtains  $H \cup U_A \subseteq \phi(E_A)$  by running Algorithm 1 on input  $\phi(E_A)$  with batches  $\mathcal{B} = \{\phi(e) : e \in E_A\}$ . She communicates  $\mathcal{R}(H \cup U_A)$  to Bob.
  3. Bob outputs the maximum weight matching in  $\mathcal{R}(H \cup U_A) \cup E_B$ .
- 

underfull edges. We will define an auxiliary fractional matching  $x$  on  $\mathcal{R}(H \cup U)$  of weight at least  $(2/3 - \epsilon)\mu_w(G)$ . We will then extend it to a fractional matching  $y$  on  $E_B \cup \mathcal{R}(H \cup U_A)$ , and show that due to the additional edges in  $E_B$ , the fractional matching  $y$  has weight at least  $(5/6 - \epsilon)\mu_w(G)$ .

Let  $E_{late} := E \setminus E_{early}$ . Fix a maximum weight matching  $M^*$  in  $E_{late}$ . Define a fractional matching  $x$  on  $\mathcal{R}(H \cup U)$  as follows:

- Start with  $H_1 = H$  and  $U_1 = U$ .
- For  $i = 1, \dots, \lambda\beta$ :
  - Let  $M_i$  be a maximum weight matching in  $\mathcal{R}(H_i \cup U_i)$ .
  - Let  $H_{i+1} = H_i \setminus \phi(M_i \setminus M^*)$  and  $U_{i+1} = U_i \setminus \phi(M_i \setminus M^*)$ .
- For every edge  $e$ , let  $x_e = \frac{|\{i: e \in M_i\}|}{\lambda\beta}$ .

In other words, we start with  $H \cup U$ , and then in each iteration, we find a maximum weight matching  $M_i$  in the refolding, and remove the edges in  $\phi(M_i \setminus M^*)$  from  $H \cup U$ .

► **Remark 5.1.** Note that this is a valid fractional matching, since

$$x_u = \sum_{e \ni u} x_e = \sum_{e \ni u} \frac{|\{i : e \in M_i\}|}{\lambda\beta} = \sum_i \frac{|\{e \ni u : e \in M_i\}|}{\lambda\beta} \leq 1.$$

Furthermore, note that  $x_e \leq \frac{1}{\lambda\beta}$  whenever  $e \notin M^*$ . This is because, if  $e \in M_i \setminus M^*$  for some  $i$ , then  $e \notin \mathcal{R}(H_j \cup U_j)$  for all  $j > i$ .

► **Lemma 5.2.** *It holds that  $\sum_e w_e x_e \geq (\frac{2}{3} - \epsilon)\mu_w(E_{late})$ .*

**Proof.** For each  $i$ , let  $G_i := E_{late} \setminus (\cup_{j < i} M_j \setminus M^*)$ . We will apply Lemma 4.13 to the graph  $G \setminus (\cup_{j < i} M_j \setminus M^*)$  and subgraph  $G_S = G_i$ . Recall that we obtain  $H_{i+1}$  from  $H_i$  by removing the edges in  $\phi(M_i \setminus M^*)$ . Since  $\phi(M_i \setminus M^*)$  is a matching, the degree of each edge in  $\phi(G)$  will decrease by at most two in each iteration. Therefore,  $U_i$  contains all the edges in  $G_i \setminus H_i$  that have  $H_i$  degree less than  $(1 - \lambda)\beta - 2(i - 1) \geq (1 - 3\lambda)\beta$ . By Lemma 4.13, we get

$$w(M_i) = \mu_w(\mathcal{R}(H_i \cup U_i)) \geq \left(\frac{2}{3} - \epsilon\right) \mu_w(G_i). \quad (5)$$

Also,  $G_i$  is constructed so that it always contains  $M^*$ , so

$$\mu_w(G_i) \geq w(M^*) = \mu_w(E_{late}). \quad (6)$$



Combining, we obtain

$$\begin{aligned}
\sum_{e \in \mathcal{R}(H \cup U)} w_e x_e &= \sum_{e \in \mathcal{R}(H \cup U)} w_e \frac{|\{i : e \in M_i\}|}{\lambda\beta} \\
&= \frac{1}{\lambda\beta} \sum_i \sum_{e \in \mathcal{R}(H \cup U)} w_e \mathbb{1}\{e \in M_i\} \\
&= \frac{1}{\lambda\beta} \sum_i w(M_i) \\
&\geq \frac{1}{\lambda\beta} \sum_i \left(\frac{2}{3} - \epsilon\right) \mu_w(G_i). && \text{by Equation 5} \\
&\geq \left(\frac{2}{3} - \epsilon\right) \mu_w(E_{late}), && \text{by Equation 6.} \quad \blacktriangleleft
\end{aligned}$$

Recall that the set of edges that Bob has access to is  $E_B \cup \mathcal{R}(H \cup U)$ . We need to show that  $\mu_w(E_B \cup \mathcal{R}(H \cup U)) \geq (\frac{5}{6} - \epsilon)\mu_w(G)$ . We will do this by extending the fractional matching  $x$  on  $\mathcal{R}(H \cup U)$  to a fractional matching  $y$  on  $E_B \cup \mathcal{R}(H \cup U)$ . In order to describe  $y$ , we will condition on the set of early edges  $E_{early}$ , thereby fixing  $\mathcal{R}(H \cup U)$  and  $x$ . For each edge  $e \in E_{late}$ , we have

$$\Pr[e \in E_B | e \in E_{late}] = \frac{\Pr[e \in E_B \wedge e \in E_{late}]}{\Pr[e \in E_{late}]} = \frac{p}{1 - \epsilon/W}.$$

and

$$\Pr[e \in E_A | e \in E_{late}] = 1 - \frac{p}{1 - \epsilon/W}.$$

Recall that  $M^*$  is a fixed maximum weight matching in  $E_{late}$ . Let  $M_{in} := M^* \cap \mathcal{R}(H \cup U)$  and let  $M_{out} := M^* \setminus \mathcal{R}(H \cup U)$ . After drawing  $E_B$ , define a random matching  $M' \subseteq M^*$  as follows:

- Include each edge  $e \in M_{in}$  independently with probability  $p$ .
- Include each edge  $e \in M_{out} \cap E_B$  independently with probability  $1 - \epsilon/W$ .

Conditioned on  $E_{early}$ , each edge in  $M_{out}$  ends up in  $M'$  independently with probability  $(1 - \epsilon/W) \cdot \frac{p}{1 - \epsilon/W} = p$ . Each edge in  $M_{in}$  also ends up in  $M'$  independently with probability  $p$ , so overall each edge in  $M^*$  ends up in  $M'$  independently with probability  $p$ .

For any edge  $e \notin M^*$ , let  $p_e$  denote the probability that  $e$  is *not* adjacent to any edge in  $M'$ . In other words,

$$p_e = \begin{cases} (1 - p) & \text{if } e \text{ has exactly one endpoint matched by } M^*, \\ (1 - p)^2 & \text{if both of the endpoints of } e \text{ are matched by } M^*. \end{cases}$$

We can now define  $\hat{y}$  on  $E_B \cup \mathcal{R}(H \cup U)$ .

$$\hat{y}_e = \begin{cases} 1 & \text{if } e \in M', \\ x_e & \text{if } e \in M^* \setminus M', \\ 0 & \text{if } e \notin M^* \text{ and } e \text{ is adjacent to at least one edge of } M' \\ (1 - p) \frac{x_e}{p_e} & \text{if } e \notin M^* \text{ and } e \text{ is not adjacent to } M'. \end{cases}$$

Finally, we scale down  $\hat{y}$  and zero out some of the entries in order to obtain a valid fractional matching  $y$ .

$$y_{(u,v)} = \begin{cases} 0 & \text{if } \hat{y}_u/(1 + \epsilon) > 1 \text{ or } \hat{y}_v/(1 + \epsilon) > 1 \\ \frac{\hat{y}_{(u,v)}}{1 + \epsilon} & \text{otherwise.} \end{cases}$$

► **Lemma 5.3.** *It holds that*

$$\mathbb{E} \left[ \sum_{e \in E} w_e y_e \right] \geq \left( \frac{2}{3} + \frac{p}{3} - 4\epsilon \right) \mu_w(G).$$

The proof is similar to Lemma 4.6 in [20], and is included in the full version of the paper. Next, we round  $y$  to an integral matching.

► **Lemma 5.4.** *There exists a matching of weight at least  $(1 - 3\epsilon) \sum_{e \in E} w_e y_e$  in  $E_B \cup \mathcal{R}(H \cup U_A)$ .*

The proof is similar to Lemma 4.7 in [20] and is included in the full version of the paper. Finally, we show that we have a large matching with high probability, and not just in expectation.

► **Lemma 5.5.** *With probability at least  $1 - n^{-5}$ , there exists a matching of weight at least  $(\frac{2}{3} + \frac{p}{3} - O(\epsilon)) \mu_w(G)$  in  $E_B \cup \mathcal{R}(H \cup U_A)$ .*

The proof is similar to Lemma 5.2 in [20], and is included in the full version of the paper. We now complete the proofs of Theorem 1.2 and Theorem 1.3.

**Proof of Theorem 1.2.** Suppose that the edge weights are in  $[W]$ . By Proposition 4.4, Protocol 2 uses  $O(n \log n \text{poly}(W/\epsilon))$  words of communication with high probability. By Lemma 5.5, the protocol achieves a  $(\frac{2}{3} + \frac{p}{3} - O(\epsilon))$ -approximation with high probability. So by Theorem 4.1, there exists a protocol that achieves a  $(\frac{2}{3} + \frac{p}{3} - O(\epsilon))(1 - \epsilon)$ -approximation using space  $O(n \log n \log R)$  when the edge weights are in  $\mathbb{R}^+$ . Letting  $p = \frac{1}{2}$  and re-scaling  $\epsilon$  proves the theorem. ◀

**Proof of Theorem 1.3.** Suppose that the edge weights are in  $[W]$ . We need to adjust the protocol to the  $k$ -party model. The first party simulates the start of a random-order stream by selecting an ordering of their edges uniformly at random. They unfold the edges and run Algorithm 1 on the corresponding unweighted  $W$ -batch random-order stream. They pass the memory state of the algorithm to the next party. Each of the next  $k - 2$  parties will continue to simulate the random-order stream that way. The  $(k - 1)$ st party communicates  $\mathcal{R}(H \cup U)$  to the last party, where  $H \cup U$  is the unweighted graph computed by Algorithm 1 on the unfolded  $W$ -batch stream. Finally, the last party will output the maximum weight matching in the graph consisting of all edges to which they have access. That way, we can set  $p = \frac{1}{k}$  and treat the first  $k - 1$  parties as Alice and the last party as Bob. By Proposition 4.4, the protocol uses  $O(n \log n \text{poly}(W/\epsilon))$  words of communication with high probability. By Lemma 5.5, the protocol achieves a  $(\frac{2}{3} + \frac{p}{3} - O(\epsilon))$ -approximation with high probability. So by Theorem 4.1, there exists a protocol that achieves a  $(\frac{2}{3} + \frac{p}{3} - O(\epsilon))(1 - \epsilon)$ -approximation using space  $O(n \log n \log R)$  when the weights are in  $\mathbb{R}^+$ . Re-scaling  $\epsilon$  proves the theorem. ◀

---

## References

- 1 Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.
- 2 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.
- 3 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In *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.

- 4 Sepehr Assadi. A simple  $(1 - \epsilon)$ -approximation semi-streaming algorithm for maximum (weighted) matching. In *2024 Symposium on Simplicity in Algorithms, SOSA 2024, Alexandria, VA, USA, January 8-10, 2024*, pages 337–354. SIAM, 2024.
- 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 *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, CA, USA, January 6-9, 2019*, pages 1616–1635, 2019.
- 6 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 7 Sepehr Assadi and Soheil Behnezhad. On the Robust Communication Complexity of 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 48:1–48:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 8 Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 131–144. ACM, 2023.
- 9 Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, pages 11:1–11:20, 2019.
- 10 Sepehr Assadi, Arun Jambulapati, Yuja Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *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.
- 11 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, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742. SIAM, 2017.
- 12 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *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.
- 13 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 *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 354–364. IEEE, 2020.
- 14 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 2021, Virtual Conference, January 11-12, 2021*, pages 165–171. SIAM, 2021.
- 15 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021.
- 16 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 342–353. IEEE, 2020.

## 16:22 Weighted Matching in Random-Order Streams

- 17 Sepehr Assadi and Vihan Shah. An asymptotically optimal algorithm for maximum matching in dynamic streams. In *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.
- 18 Sepehr Assadi and Janani Sundaresan. Hidden permutations to the rescue: Multi-pass streaming lower bounds for approximate matchings. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 909–932. IEEE, 2023.
- 19 Sepehr Assadi and Janani Sundaresan. (Noisy) gap cycle counting strikes back: Random order streaming lower bounds for connected components and beyond. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 183–195. ACM, 2023.
- 20 Amir Azarmehr and Soheil Behnezhad. Robust communication complexity of matching: EDCS achieves  $5/6$  approximation. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 14:1–14:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 21 Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching with few queries: New algorithms and tools. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, CA, USA, January 6-9, 2019*, pages 2855–2874. SIAM, 2019.
- 22 Aaron Bernstein. Improved bounds for matching in random-order streams. In *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.
- 23 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 668–681. ACM, 2021.
- 24 Aaron Bernstein and Cliff Stein. Fully Dynamic Matching in Bipartite Graphs. In *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 167–179. Springer, 2015.
- 25 Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 692–711. SIAM, 2016.
- 26 Stéphane Boucheron, Gábor Lugosi, and Pacal Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14:1884–1899, 2009.
- 27 Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural results on matching estimation with applications to streaming. *Algorithmica*, 81(1):367–392, 2019.
- 28 Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 2015.
- 29 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, STOC '08, pages 641–650. ACM, 2008.
- 30 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 570–583. ACM, 2021.

- 31 Ashish Chiplunkar, John Kallaugher, Michael Kapralov, and Eric Price. Factorial lower bounds for (almost) random order streams. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 486–497. IEEE, 2022.
- 32 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 2016, Arlington, VA, USA, January 10–12, 2016*, pages 1326–1344. SIAM, 2016.
- 33 Rajesh Hemant Chitnis, Graham Cormode, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4–6, 2015*, pages 1234–1251. SIAM, 2015.
- 34 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 35 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 2014, September 4–6, 2014, Barcelona, Spain*, volume 28 of *LIPICs*, pages 96–104. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.
- 36 Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In *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.
- 37 Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, 2012.
- 38 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.
- 39 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.
- 40 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12–15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016.
- 41 Alireza Farhadi, MohammadTaghi Hajiaghayi, Tung Mah, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5–8, 2020*, pages 1773–1785. SIAM, 2020.
- 42 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
- 43 Moran Feldman and Ariel Szarf. Maximum matching sans maximal matching: A new approach for finding maximum matchings in the data stream model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19–21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPICs*, pages 33:1–33:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

- 44 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 *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 – 24, 2022*, pages 248–260. ACM, 2022.
- 45 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *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.
- 46 Paritosh Garg, Sagar Kale, Lars Rohwedder, and Ola Svensson. Robust Algorithms Under Adversarial Injections. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 56:1–56:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 47 Mohsen Ghaffari and David Wajc. Simplified and Space-Optimal Semi-Streaming  $(2+\epsilon)$ -Approximate Matching. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASICs*, pages 13:1–13:8. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 48 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 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012.
- 49 Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 273–279. ACM, 2006.
- 50 Sudipto Guha and Andrew McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wrocław, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 704–715. Springer, 2007.
- 51 Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM J. Comput.*, 38(5):2044–2059, 2009.
- 52 Manoj Gupta and Richard Peng. Fully dynamic  $(1+\epsilon)$ -approximate matchings. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 548–557. IEEE Computer Society, October 2013.
- 53 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016.
- 54 Chien-Chung Huang and François Sellier. Maximum Weight  $b$ -Matchings in Random-Order Streams. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPICs*, pages 68:1–68:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 55 Shang-En Huang and Hsin-Hao Su.  $(1-\epsilon)$ -approximate maximum weighted matching in  $\text{poly}(1/\epsilon, \log n)$  time in the distributed and parallel settings. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023*, pages 44–54. ACM, 2023.
- 56 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 57 Ming-Yang Kao, Tak-Wah Lam, Wing-Kin Sung, and Hing-Fung Ting. A decomposition theorem for maximum weight bipartite matchings. *SIAM Journal on Computing*, 31(1):18–26, 2001.

- 58 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697. SIAM, 2013.
- 59 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. SIAM, 2021.
- 60 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 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751. SIAM, 2014.
- 61 Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1753–1772. SIAM, 2020.
- 62 Christian Konrad. Maximum matching in turnstile streams. In *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.
- 63 Christian Konrad. A Simple Augmentation Method for Matchings with Applications to Streaming Algorithms. In *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.
- 64 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. Springer, 2012.
- 65 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. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 66 Christian Konrad and Kheeran K. Naidu. An unconditional lower bound for two-pass streaming algorithms for maximum matching approximation. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 2881–2899. SIAM, 2024.
- 67 Christian Konrad, Kheeran K. Naidu, and Arun Steward. Maximum matching via maximal matching queries. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPICs*, pages 41:1–41:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 68 Harry Lang. Online facility location on semi-random streams. *CoRR*, abs/1711.09384, 2017. [arXiv:1711.09384](https://arxiv.org/abs/1711.09384).
- 69 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 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005.
- 70 Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPICs*, pages 17:1–17:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.


- 71 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 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASICs*, pages 14:1–14:4. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 72 Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPICs*, pages 131:1–131:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 73 Ami Paz and Gregory Schwartzman. A  $(2 + \epsilon)$ -approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, January 16-19*, pages 2153–2161. SIAM, 2017.
- 74 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.
- 75 Sumedh Tirodkar. Deterministic algorithms for maximum matching on general graphs in the semi-streaming model. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, volume 122 of *LIPICs*, pages 39:1–39:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 76 David Wajc. Negative association-definition, properties, and applications, 2017.
- 77 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1979, Atlanta, Georgia, USA*, pages 209–213. ACM, 1979.
- 78 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.




# Competitive Query Minimization for Stable Matching with One-Sided Uncertainty

**Evrpidis Bampis** ✉ 

Sorbonne Université, CNRS, LIP6, Paris, France

**Konstantinos Dogeas** 

Department of Computer Science, Durham University, Durham, United Kingdom

**Thomas Erlebach** ✉ 


Department of Computer Science, Durham University, Durham, United Kingdom

**Nicole Megow** ✉ 

Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany

**Jens Schlöter** ✉ 

Faculty of Mathematics and Computer Science, University of Bremen, Bremen, Germany

**Amitabh Trehan** ✉ 

Department of Computer Science, Durham University, Durham, United Kingdom

---

## Abstract

We study the two-sided stable matching problem with one-sided uncertainty for two sets of agents  $A$  and  $B$ , with equal cardinality. Initially, the preference lists of the agents in  $A$  are given but the preferences of the agents in  $B$  are unknown. An algorithm can make queries to reveal information about the preferences of the agents in  $B$ . We examine three query models: comparison queries, interviews, and set queries. Using competitive analysis, our aim is to design algorithms that minimize the number of queries required to solve the problem of finding a stable matching or verifying that a given matching is stable (or stable and optimal for the agents of one side). We present various upper and lower bounds on the best possible competitive ratio as well as results regarding the complexity of the offline problem of determining the optimal query set given full information.

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

**Keywords and phrases** Matching under Preferences, Stable Marriage, Query-Competitive Algorithms, Uncertainty

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.17

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.10170> [3]

**Funding** This research was supported by EPSRC grant EP/S033483/2.

*Evrpidis Bampis*: Partially funded by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR).

*Nicole Megow*: Supported by DFG grant no. 517912373.

## 1 Introduction

In the classical two-sided stable matching problem, we are given two disjoint sets  $A$  and  $B$  of agents (often referred to as men and women) of equal cardinality  $n$ . Each agent has a complete preference list over the agents of the other set. The task is to find a *stable* matching, i.e., a one-to-one allocation in which no two agents prefer to be matched to each other rather than to their current matching partners. This problem has applications in numerous allocation markets, e.g., university admission, residency markets, distributed



© Evripidis Bampis, Konstantinos Dogeas, Thomas Erlebach, Nicole Megow, Jens Schlöter, and Amitabh Trehan;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 17; pp. 17:1–17:21



Leibniz International Proceedings in Informatics

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

internet services, etc. Since its introduction by Gale and Shapley [13] this problem has been widely studied in different variants from both practical and theoretical perspectives; we refer to the books [15, 31, 25].

While the majority of the literature assumes full information about the preference lists, this may not be realistic in large matching markets. It might be impractical or too costly and not even necessary to gather the complete preferences. Hence, different models for uncertainty in the preferences have received attention in the past decade [1, 2, 6, 7, 9, 17, 16, 29, 30]. Many of these works rely on probabilistic models and guarantees. This may not be appropriate for applications in which no (correct) distributional information is available, e.g. in one-time markets. Further, one might ask for guaranteed properties such as stability and optimality instead of probabilistic ones.

A different way of handling uncertainty in the preferences is to allow an algorithm to make queries to learn about the unknown preferences. Various types of queries (in terms of both input and output) are conceivable, with one example being *interview queries* [6, 7, 29, 30]. Here one asks for a query sequence where a query corresponds to an interview between two potential matching partners and the outcome is the placement of the interview partners in each other's preference list among all other candidates that she has interviewed so far. Hence, if an agent has several such interviews then she finds out her preference order over all these candidates.

In this paper we investigate various query models for stable matching problems with *one-sided* uncertainty in the preferences. We assume that initially only the preference lists of one side,  $A$ , are known but the preference lists of the other side,  $B$ , are unknown. Applications include allocations between groups of different seniority or when preferences shall be kept private; see also [17, 16, 18]. For illustration consider, e.g., pairing new staff with mentors or new PhD students with supervisors as part of the onboarding. New staff can be asked to provide a full preference list of mentors based on information about the available mentors that can be made accessible with little effort, while requiring mentors to rank potential mentees might be considered too burdensome for senior staff due to other significant time commitments.

We consider three types of queries to gain information about the preferences, namely (i) *comparison queries* that reveal for an agent  $b \in B$  and a pair of agents from  $A$  which one  $b$  prefers, (ii) *set queries* that reveal for an agent  $b \in B$  and a subset  $S \subseteq A$  the agent in  $S$  that  $b$  prefers most, and (iii) *interview queries*.

We study basic problems regarding stability and optimality of matchings using these query models. A stable matching is called  $A$ -optimal (resp.  $B$ -optimal) if no agent in  $A$  (resp.  $B$ ) prefers a different stable matching over the current one. To our knowledge, most existing related work considers worst-case bounds on the absolute number of queries necessary to solve the respective problem; see Further Related Work below for a discussion. For many instances, however, executing such a worst-case number of queries might not be necessary. To also optimize the number of queries on these instances, we analyze our algorithms using *competitive analysis*. We say that an algorithm that makes queries until it can output a provably correct answer, e.g., a stable and  $A$ -optimal matching, is  $\rho$ -competitive (or  $\rho$ -query-competitive) if it makes at most  $\rho$  times as many queries as the minimum possible number of queries that also output a provably correct answer for the given instance. Note that this answer may differ from that of the algorithm, e.g., a different stable matching. In this paper, we design upper and lower bounds on the competitive ratios for the above mentioned problems and query models. Our results illustrate that worst-case instances regarding the competitive ratio are very different from the worst-case instances regarding the absolute number of queries. Thus,

our lower bounds on the competitive ratio use different instances and our algorithms are designed to optimize on different instances. Indeed, worst-case instances for the absolute number of queries turn out to be “easy” for competitive analysis as the optimal solution we compare against is very large.

Query-competitive algorithms are often associated with the field of “explorable uncertainty”. Most previous work considers queries revealing an originally uncertain *value* [4, 8, 10, 19, 20, 22, 26, 11], while in this work we query a *preference*.

**Our Contribution.** We study the stable matching problem with one-sided uncertainty in the preference lists and give the following main results. Note that we assume that the preferences of the  $B$  side are unknown, and  $|A| = |B| = n$ . We remark that our technically most involved main results are lower bounds on the competitive ratio and hardness results, so the results only get stronger by making these assumptions.

In Section 3 we focus on *comparison queries*. Firstly, we ask the question of how to verify that a given matching is stable. We show that the problem can be solved with a 1-competitive algorithm. Then we ask how to find a stable matching under one-sided uncertainty. We give a 1-competitive algorithm that finds a stable matching and, moreover, the solution is provably  $A$ -optimal. Essentially, we employ the well-known *deferred acceptance algorithm*, first analyzed by Gale and Shapley [13], and compare its number of queries carefully with the number of queries that any algorithm needs to verify a stable matching.

A substantially more challenging task is to find a  $B$ -optimal stable matching. Note that a trivial competitive ratio is  $O(n^2 \log n)$ , as it is possible to obtain the full preferences of each of the  $n$  elements in  $B$  using  $O(n \log n)$  queries, and the optimum total number of queries is at least 1. One of our main contributions is a tight bound of  $O(n)$ . To that end, we first show that every algorithm for verifying that a given matching is  $B$ -optimal and stable requires  $\Omega(n)$  queries. Then we give an  $O(n)$ -competitive algorithm for the problem of finding one. This is best possible up to constant factors, which we prove with a matching lower bound that also holds for verifying that a given matching is stable and  $B$ -optimal, even for randomized algorithms.

We complement these results by showing that the offline problem of determining the optimal number of queries for finding the  $B$ -optimal stable matching is NP-hard, and we give an  $O(\log n \log \log n)$ -approximation algorithm. Here, the *offline* version of a problem is to compute, given full information about the preferences of all agents, a smallest set of queries with the property that an algorithm making exactly those queries has sufficient information to solve the problem with one-sided uncertainty.

Section 4 discusses interview queries. We show that the bounds on the competitive ratio and hardness results for comparison queries translate to interview queries. We remark that some of these results for interview queries, e.g., a 1-competitive algorithm for finding an  $A$ -optimal stable matching, were already proven by Rastegari et al. [30] and discuss differences to their results in the corresponding section. Interestingly, we can use essentially the same techniques as for the comparison model. This may seem surprising, especially for the lower bounds, as interview queries seem to be more powerful. For instance,  $n$  interviews are sufficient to determine the precise preference order of an agent  $b \in B$ , while we need  $\Omega(n \log n)$  comparison queries to determine  $b$ 's preference order. On the other hand, an instance that can be solved with a single comparison query requires two interviews. In general, we can simulate a comparison query by using two interview queries.

In Section 5 we discuss the *set query* model. While some bounds remain the same as in the other models, e.g., 1-competitiveness for verifying the stability of a given matching, we show that some bounds change drastically. For example, we give an  $O(\log n)$ -competitive

algorithm for verifying that a given matching is  $B$ -optimal, which is in contrast to the lower bound of  $\Omega(n)$  in the other query models. It remains open whether  $\mathcal{O}(1)$ -competitive algorithms exist for the problems of finding a stable matching or verifying a  $B$ -optimal matching with set queries.

**Further Related Work.** In classical work on stable matching with queries, the preferences on both sides can only be accessed via queries, with a query usually either asking for the  $i$ th entry in a preference list or for the rank of a specific element within a preference list (cf. e.g. [27]). Note that two rank queries are sufficient to simulate a comparison query, but up to  $n - 1$  comparison queries are needed to obtain the information of a single rank query. Thus, existing lower bounds on the necessary number of rank queries in these query models translate to our setting (up to a constant factor), but upper bounds do not necessarily translate. Ng and Hirschberg [27] showed that  $\Theta(n^2)$  such queries are necessary to find or verify a stable matching in the worst case. The lower bound of  $\Omega(n^2)$  translates to any type of queries with boolean answers, including comparison queries [14]. Further work on interview queries includes empirical results [6, 7] and complexity results [29] on several decision problems under partial uncertainty. We discuss the latter in Section 4.

Our setting of one-sided uncertainty and querying uncertain preferences is also related to existing work on online algorithms for *eliciting partial preferences* [21, 28, 24]. These works also consider a setting where the preferences of agents in one of the sets are uncertain but can be determined by using different types of queries. In particular, [28] also considers the set query model. The main difference to our work is that these papers assume that the elements of one set do not have any preferences at all. As a consequence, they do not consider stability at all and instead aim at computing pareto-optimal or rank-maximal matchings.

## 2 Preliminaries

An instance of the *two-sided stable matching problem* consists of two disjoint sets  $A$  and  $B$  of size  $|A| = |B| = n$  and complete preference lists: The preference list for each agent  $a \in A$  is a total order  $\prec_a$  of  $B$ , the preference list of each agent  $b \in B$  is a total order  $\prec_b$  of  $A$ . Here,  $a_1 \prec_b a_2$  means that  $b$  prefers  $a_1$  to  $a_2$ . A matching is a bijection from  $A$  to  $B$ . For a matching  $M$ , we denote the element of  $B$  that is matched to  $a \in A$  by  $M(a)$ , and the element of  $A$  that is matched to  $b \in B$  by  $M(b)$ .

Given a matching  $M$ , a pair  $(a, b) \in A \times B$  is a *blocking pair* in  $M$  if  $a$  is not matched to  $b$  in  $M$ ,  $a$  prefers  $b$  to  $M(a)$ , and  $b$  prefers  $a$  to  $M(b)$ . A matching  $M$  is called a *stable matching* if there is no blocking pair in  $M$ .

In their influential paper, Gale and Shapley [13] showed that a stable matching always exists, and the *deferred acceptance algorithm* computes one in  $\mathcal{O}(n^2)$  time. In this algorithm, one group ( $A$  or  $B$ ) proposes matches and the other decides whether to accept or reject each proposal. The algorithm produces a stable matching that is best possible for the group  $X$  that proposes (we say  $X$ -optimal) and worst possible for the other group: Each element of the group that proposes gets matched to the highest-preference element to which it can be matched in any stable matching, and each element of the other group gets matched to the lowest-preference element to which it can be matched in any stable matching.

In this paper, we consider the setting of *one-sided uncertainty*, where initially only the preference lists of all agents in  $A$  are known, but the preference lists of  $b \in B$  are unknown. An algorithm can make queries to learn about the preferences of  $b \in B$ . We distinguish the following types of queries:

- *Comparison queries:* For agents  $b \in B$  and  $a_1, a_2 \in A$ , the query  $\text{prefer}(b, a_1, a_2)$  returns  $a_1$  if  $b$  prefers  $a_1$  to  $a_2$  and  $a_2$  otherwise. These queries can also be seen as Boolean queries that return true iff  $b$  prefers  $a_1$  to  $a_2$ .
- *Set queries:* For agents  $b \in B$  and any subset  $S \subseteq A$ , the query  $\text{top}(b, S)$  returns  $b$ 's most preferred element of  $S$ .
- *Interview queries:* For agents  $b \in B$  and  $a \in A$ , an interview query  $\text{intq}(b, a)$  reveals the total order of the subset  $\{a\} \cup P_b$  defined by  $\prec_b$ , where  $P_b$  is the set of all elements  $a' \in A$  for which a query  $\text{intq}(b, a')$  has already been executed before the query  $\text{intq}(b, a)$ .

A *stable matching instance with one-sided uncertainty* is given by two sets  $A$  and  $B$  of size  $n$  and, for each agent  $a \in A$ , a total order  $\prec_a$  of the agents in  $B$ . The preferences of the agents in  $B$  are initially unknown. For a given stable matching instance with one-sided uncertainty, we consider the following problems: *finding a stable matching*, *finding an  $A$ -optimal stable matching*, and *finding a  $B$ -optimal stable matching*. For a given stable matching instance with one-sided uncertainty and a matching  $M$ , we consider the following problems: *verifying that  $M$  is stable*, *verifying that  $M$  is stable and  $A$ -optimal*, and *verifying that  $M$  is stable and  $B$ -optimal*. All problems can be considered for each query model. For the verification problems, we consider the competitive ratio only for inputs where  $M$  is indeed a stable (and  $A$ - or  $B$ -optimal) matching. If this is not the case, the algorithm must detect this, but we do not compare the number of queries it makes to the optimum. This is because any algorithm may be required to make up to  $\Omega(n^2)$  comparison or interview queries to detect a blocking pair, while the optimum can prove its existence with a constant number of queries.

It is easy to see that for the optimum, the problem of verifying that a given matching  $M$  is stable and  $A$ -optimal ( $B$ -optimal) is the same as that of finding the  $A$ -optimal ( $B$ -optimal) stable matching. This implies that any lower bound on the number of queries required to verify that  $M$  is stable and  $A$ -optimal ( $B$ -optimal) also applies to the problem of finding the  $A$ -optimal ( $B$ -optimal) stable matching.

An important concept is the notion of *rotations*, which can be defined as follows (cf. [25]): Let a stable matching  $M$  be given. For an agent  $a_i \in A$ , let  $s_A(a_i)$  denote the most-preferred element  $b_j$  on  $a_i$ 's preference list such that  $b_j$  prefers  $a_i$  to her current partner  $M(b_j)$ . Note that  $s_A(a_i)$  must be lower than  $M(a_i)$  in  $a_i$ 's preference list as otherwise  $(a_i, s_A(a_i))$  would be a blocking pair. Let  $\text{next}_A(a_i) = M(s_A(a_i))$ . Then a rotation (exposed) in  $M$  is a sequence  $(a_{i_0}, b_{j_0}), \dots, (a_{i_{r-1}}, b_{j_{r-1}})$  of pairs such that, for each  $k$  ( $0 \leq k \leq r-1$ ),  $(a_{i_k}, b_{j_k}) \in M$  and  $a_{i_{k+1}} = \text{next}_A(a_{i_k})$ , where addition is modulo  $r$ . The rotation can be viewed as an alternating cycle consisting of the matched edges  $(a_{i_k}, b_{i_k})$  and the unmatched edges  $(a_{i_k}, b_{i_{k+1}})$  (for  $0 \leq k \leq r-1$ ). We refer to an edge  $(a, s_A(a))$  as a *rotation edge* or  *$r$ -edge* as it can potentially be part of a rotation. Note that every vertex  $a \in A$  is incident with at most one  $r$ -edge.

Given a rotation  $R$  in a stable matching  $M$ , we can construct a stable matching  $M'$  from  $M$  by removing all edges that are part of  $R$  and  $M$  and adding all  $r$ -edges that are part of  $R$ . We refer to this as *applying* a rotation. Observe that no agent in  $B$  is worse off in  $M'$  than in  $M$ , and some agents in  $B$  prefer  $M'$  to  $M$ . The following has been shown.

► **Lemma 1** (Lemma 2.5.3 in Gusfield and Irving [15]). *If  $M$  is any stable matching other than the  $B$ -optimal stable matching, then there is at least one rotation exposed in  $M$ .*

### 3 Stable Matching with Comparison Queries

In this section, we consider the comparison query model. We first discuss our results on the problems of verifying that a given matching is stable and finding an  $A$ -optimal matching, before moving on to our main results regarding the competitive ratio for finding/verifying a  $B$ -optimal matching.

### 3.1 Verifying That a Given Matching Is Stable

We consider the *verification problem* where we are given a matching  $M$  and our task is to verify that  $M$  is indeed stable. As argued in the previous section, we only care about the competitive ratio if the given matching  $M$  is indeed stable.

The following auxiliary lemma shows that exploiting transitivity cannot reduce the number of comparison queries to an agent  $b \in B$  if one needs to find out the preference relationship of  $k$  agents from  $A$  to one particular agent from  $A$  in  $b$ 's preference list.

► **Lemma 2.** *Consider two agents  $a \in A$ ,  $b \in B$  and assume that there are  $k$  agents  $a_1, \dots, a_k \in A \setminus \{a\}$  for each of which we want to know whether  $b$  prefers that agent to  $a$  or not. Then exactly  $k$  comparison queries to  $b$  are necessary and sufficient to obtain this knowledge.*

**Proof.** The  $k$  queries  $\text{prefer}(b, a, a_i)$  for  $i = 1, \dots, k$  are clearly sufficient. Assume that  $k'$  queries to  $b$ , for some  $k' < k$ , are sufficient to obtain the desired information. Consider the auxiliary graph  $H$  with vertex set  $V_H = A$  and an edge  $\{a', a''\}$  for each of those  $k'$  queries  $\text{prefer}(b, a', a'')$ . As the set  $A' = \{a, a_1, a_2, \dots, a_k\}$  has  $k+1$  vertices and  $H$  has fewer than  $k$  edges, the set  $A'$  intersects at least two different connected components of  $H$ . Let  $a_j$ , for some  $1 \leq j \leq k$ , be a vertex that does not lie in the same component as  $a$ . Then the  $k'$  queries do not show whether  $b$  prefers  $a_j$  to  $a$  or not, which contradicts  $k'$  queries being sufficient. ◀

If an algorithm obtains for agents  $x$  and  $y$  with  $y \neq M(x)$  the information that  $M(x) \prec_x y$  (either via a direct query or via transitivity), we say that the algorithm *relates*  $y$  to  $M(x)$  for  $x$ . By Lemma 2, if the optimum relates  $k$  different elements to  $M(x)$  for  $x$ , it needs to make  $k$  queries to  $x$ . A pair  $(x, y)$  with  $y \neq M(x)$  such that the optimum relates  $y$  to  $M(x)$  for  $x$  is called a *relationship pair* (for  $x$ ). Lemma 2 implies the following.

► **Corollary 3.** *The total number of relationship pairs (for all agents  $x$ ) is a lower bound on the number of comparison queries the optimum makes.*

► **Theorem 4.** *Given a stable matching instance with one-sided uncertainty and a stable matching  $M$ , there is a 1-competitive algorithm that uses  $\sum_{a \in A} |\{b \in B \mid b \prec_a M(a)\}|$  queries for verifying that  $M$  is stable in the comparison query model.*

**Proof.** Since the preferences of agents on the  $A$ -side are not uncertain, for each  $(a, b) \notin M$ , we already know whether  $M(a) \prec_a b$ . If  $M(a) \prec_a b$ , then we do not have to execute any queries to show that  $(a, b) \notin M$  is not a blocking pair. Otherwise, every feasible query set has to prove  $M(b) \prec_b a$ . Therefore, for each element  $b \in B$ , there is a uniquely determined number  $n_b$  of elements of  $A$  that any solution (including the optimum) must relate to  $M(b)$  for  $b$ . Let  $K = \sum_{b \in B} n_b$  be the resulting number of relationship pairs.

Our algorithm simply queries  $\text{prefer}(b, M(b), a)$  for every pair  $(a, b) \notin M$  for which  $b \prec_a M(a)$ . These are exactly  $K$  queries. As the total number of relationship pairs is  $K$ , the optimum must also make  $K$  queries (Corollary 3). Hence, our algorithm is 1-competitive. ◀

The proof of Theorem 4 implies that the stable matching that maximizes the number of queries that are required to prove stability is the  $B$ -optimal matching.

► **Corollary 5.** *The number of comparison queries needed to verify that the  $B$ -optimal matching is stable is  $\max_{M \text{ stable}} \sum_{a \in A} |\{b \in B \mid b \prec_a M(a)\}|$ .*

### 3.2 Finding an $A$ -Optimal Stable Matching

We obtain the following positive result by adapting the classical deferred acceptance algorithm [13] with  $A$  making the proposals.

► **Theorem 6.** *For a given stable matching instance with one-sided uncertainty, there is a 1-competitive algorithm for finding a stable matching in the comparison query model. The algorithm actually finds an  $A$ -optimal stable matching.*

**Proof.** We utilize the classical deferred acceptance algorithm [13] where  $A$  makes the proposals, and we assume the reader's familiarity with it. An unmatched agent  $a \in A$  makes a proposal to their preferred agent  $b \in B$  by whom it has never been rejected. If  $b$  is unmatched, then  $b$  accepts the proposal and  $a$  and  $b$  get matched. If  $b$  is currently matched to some  $a' \in A$ , the algorithm makes a query  $prefer(b, a, a')$ . If the query result is that  $b$  prefers  $a$  to  $a'$ , then  $b$  accepts  $a$ 's proposal and becomes matched to  $a$  while  $a'$  becomes unmatched. Otherwise,  $b$  rejects the proposal and remains matched to  $a'$ . The algorithm terminates if all agents in  $A$  are matched or if every unmatched agent in  $A$  has been declined by all agents in  $B$ .

We show that this algorithm makes the minimum possible number of comparison queries.

We execute the deferred acceptance algorithm with  $A$  as the proposers, so it produces an  $A$ -optimal stable matching. Consider an arbitrary agent  $b \in B$ . Assume that  $b$  gets matched to  $a \in A$  in the stable matching. Let  $A_b = \{a, a_1, a_2, \dots, a_{k_b}\}$  (for some  $0 \leq k_b < n$ ) be the set of agents of  $A$  that proposed to  $b$  during the execution of the algorithm. Note that  $|A_b| = k_b + 1$  and the algorithm has executed  $k_b$  queries to  $b$ , each for two agents of  $A_b$  (the first agent of  $A$  that proposed to  $b$  did not require a query). Observe that each of  $a_1, \dots, a_{k_b}$  gets matched with an agent of  $B$  that they rank strictly lower than  $b$  in the final matching.

We claim that no stable matching can be identified without making at least  $k_b$  queries to  $b$ . Let  $M'$  be an arbitrary stable matching. Note that  $b$  rates  $M'(b)$  at least as highly as  $a$ , because  $M$  is the worst possible matching for  $B$ . Furthermore, for each  $a_i$  with  $1 \leq i \leq k_b$ , we have that  $a_i$  rates  $b$  strictly higher than  $M'(a_i)$  because  $M$  is  $A$ -optimal and  $a_i$  rates  $M(a_i)$  strictly lower than  $b$ . Thus, for none of the pairs  $(a_i, b)$  for  $1 \leq i \leq k_b$  to be a blocking pair, the queries of any optimal query set must establish that  $b$  rates  $M'(b)$  more highly than every  $a_i$  for  $1 \leq i \leq k_b$ . This can only be achieved with at least  $k_b$  queries.

The same argument applies to each  $b \in B$ , so we have that both the optimal number of queries and the number of queries made by the algorithm are equal to  $\sum_{b \in B} k_b$ . ◀

The proof of Theorem 6 implies that, for the  $A$ -optimal stable matching  $M$ , the optimal number of queries to prove that  $M$  is stable equals the optimal number of queries to prove that  $M$  is stable and  $A$ -optimal. Hence, proving optimality comes in this case for free.

### 3.3 Finding a $B$ -Optimal Stable Matching

The problem of finding a  $B$ -optimal stable matching is substantially more challenging. While there still exists a 1-competitive algorithm in special cases, e.g., when all  $A$ -side preference lists are equivalent (see the full version [3]), this is not the case for arbitrary instances.

We first describe an algorithm that is  $\mathcal{O}(n)$ -competitive. Complementing this result, we then show that every (randomized) online algorithm has competitive ratio at least  $\Omega(n)$  for finding a  $B$ -optimal stable matching. Finally, we show that the offline problem of determining the optimal number of queries for computing a  $B$ -optimal stable matching is NP-hard and give an  $\mathcal{O}(\log n \log \log n)$ -approximation.

### 3.3.1 Algorithm for Computing a $B$ -Optimal Stable Matching

We first consider the problem of verifying that a given stable  $B$ -optimal matching is indeed stable and  $B$ -optimal. An algorithm for this problem needs to prove that  $M$  has no blocking pair and that no alternating cycle with respect to  $M$  is a rotation. For each potential blocking pair  $(a, b)$  that cannot be ruled out because of  $a$ 's preferences, such an algorithm has to prove that it is not a blocking pair using a suitable query to  $b$  as discussed in Section 3.1.

The more involved part is proving that  $M$  is  $B$ -optimal. By Lemma 1,  $M$  is  $B$ -optimal if and only if it does not expose a rotation. Based on the known  $A$ -side preferences, each edge  $(a, b)$  with  $M(a) \prec_a b$  could potentially be an  $r$ -edge. Thus, each cycle that alternates between such edges and edges in  $M$  could potentially be a rotation. An algorithm that proves  $B$ -optimality has to prove for each such alternating cycle that at least one non-matching edge  $(a, b)$  on that cycle is not an  $r$ -edge. By definition, there are two possible ways to prove that an edge  $(a, b)$  with  $M(a) \prec_a b$  is not an  $r$ -edge:

1. Query  $b$  and find out that  $b$  prefers  $M(b)$  to  $a$ . Then,  $b$  cannot be  $s_A(a)$  as  $b$  does not prefer  $a$  to  $M(b)$ .
2. Query one  $b'$  with  $M(a) \prec_a b' \prec_a b$  and find out that  $b'$  prefers  $a$  to  $M(b')$ . Then,  $b$  cannot be the most-preferred element in  $a$ 's list that prefers  $a$  to her current partner, as  $b'$  has that property and is preferred over  $b$ .

Corollary 5 gives the optimal number of queries to prove that the matching  $M$  is stable, which is a lower bound on the optimal number of queries necessary to prove that  $M$  is stable and  $B$ -optimal. Let  $Q(M)$  denote this number. However, there exist instances where  $Q(M) = 0$  and  $Q_B(M) > 0$  for the optimal number  $Q_B(M)$  of queries to prove that  $M$  is stable and  $B$ -optimal. Consider an instance where all elements of  $A$  have distinct first choices and let  $M$  denote the matching that matches all elements of  $A$  to their respective first choice. Then, there is a realization of  $B$ -side preference lists such that the matching  $M$  is also  $B$ -optimal. For this realization we have  $Q(M) = 0$  and  $Q_B(M) > 0$ . This implies that the lower bound of Corollary 5 is not strong enough for analyzing algorithms that verify  $B$ -optimality as we cannot prove that such an algorithm makes at most  $c \cdot Q(M)$  queries. We give another lower bound on the optimal number of queries.

► **Lemma 7.** *The optimal number of queries for verifying (and thus also for finding) the  $B$ -optimal stable matching is at least  $n - 1$  for every instance of the stable matching problem with one-sided uncertainty.*

**Proof.** Let  $M$  be the  $B$ -optimal stable matching for the given instance. For  $a \in A$ , call a query an  $a$ -query if it reveals for some  $b \in B$  with  $b \neq M(a)$  whether  $b$  prefers  $a$  to her current partner or not. We claim that an optimal algorithm needs to make at least one  $a$ -query for every  $a \in A$  with at most a single exception. Assume for a contradiction that the optimal algorithm makes neither an  $a$ -query nor an  $a'$ -query for two distinct elements  $a, a' \in A$ . If  $a$  prefers  $M(a)$  over  $M(a')$  and  $a'$  prefers  $M(a')$  over  $M(a)$ , then it is impossible to exclude the possibility that  $(a, M(a)), (a', M(a'))$  is a rotation exposed in  $M$ , because the only way to prove that  $(a, M(a'))$  is not an  $r$ -edge is via an  $a$ -query, and similarly for  $(a', M(a))$ . If  $a$  prefers  $M(a')$  over  $M(a)$ , then an  $a$ -query to  $M(a')$  is necessary to exclude that  $(a, M(a'))$  is a blocking pair. If  $a'$  prefers  $M(a)$  over  $M(a')$ , then an  $a'$ -query to  $M(a)$  is necessary for the analogous reason. Hence, the claim holds. We note that the  $n - 1$  queries whose existence is asserted by the claim are distinct: A query to some  $b \in B$  cannot be an  $a$ -query and at the same time an  $a'$ -query for some  $a' \neq a$ , as the query  $prefer(b, a, a')$  cannot yield previously unknown information about how both  $a$  and  $a'$  compare to  $M(b)$  in  $b$ 's preference list. ◀



■ **Algorithm 1** Algorithm to find the  $B$ -optimal stable matching using comparison queries.

---

**Input:** Instance of the stable matching problem with one-sided uncertainty.

```

1  $M \leftarrow A$ -optimal matching computed using Theorem 6 ;
2  $N \leftarrow \{a \in A \mid M(a) \text{ is last in } \prec_a\}$  ;          /* Elements without  $r$ -edge */
3  $\forall a \in A \setminus N: p(a) \leftarrow$  first element in  $\prec_a$  after  $M(a)$ ;
4  $\forall a \in A \setminus N: r(a) \leftarrow \top$  ;    /* known  $r$ -edges or  $\top$  if  $r$ -edge still unknown */
5 foreach  $a \in A \setminus N$  do
6   repeat
7      $t \leftarrow \text{prefer}(p(a), a, M(p(a)))$  ;
8     if  $t = M(p(a))$  then
9       if  $p(a)$  is the last element of  $\prec_a$  then  $N \leftarrow N \cup \{a\}$  ;
10      else  $p(a) \leftarrow$  direct successor of  $p(a)$  in  $\prec_a$  ;
11      else
12         $r(a) \leftarrow p(a)$ ;          /*  $r(a)$  and  $a$  form an  $r$ -edge */
13    until  $r(a) \neq \top$  or  $a \in N$ ;
14 if  $M$  exposes a rotation  $R$  then
15    $M \leftarrow$  stable matching constructed from  $M$  by applying  $R$ ;
16    $N \leftarrow N \cup \{a \in A \cap R \mid M(a) \text{ is last in } \prec_a\}$ ;
17    $\forall a \in (A \cap R) \setminus N: r(a) \leftarrow \top$  and  $p(a) \leftarrow$  first element in  $\prec_a$  after  $M(a)$ ;
18    $\forall a \in (A \setminus R) \setminus N: p(a) \leftarrow r(a)$  and  $r(a) \leftarrow \top$ ;
19   Jump to Line 5;
20 return  $M$ ;

```

---

Next, we give an  $\mathcal{O}(n)$ -competitive algorithm for finding a  $B$ -optimal matching and analyze it by exploiting the lower bounds on the optimal number of queries of Corollary 5 and Lemma 7. For pseudocode see Algorithm 1.

1. Find an  $A$ -optimal matching using the 1-competitive algorithm for  $A$ -optimal matchings.
2. Search for a rotation by asking, for every  $a \in A$ , the elements of  $B$  that are below  $M(a)$  in  $a$ 's preference list in order of  $\prec_a$  whether they prefer  $a$  to their current partner, until either an  $r$ -edge is found or we know that  $a$  has no  $r$ -edge.
3. If a rotation  $R$  is found, apply that rotation. The agents  $a \in A \cap R$  then no longer have a known  $r$ -edge as their previous  $r$ -edge is now their matching edge. However, the new  $r$ -edge partner of such an agent must be further down the preference list of  $a$  than the old one. The elements  $a \in A \setminus R$  that had an  $r$ -edge to an element  $b \in B \cap R$  can no longer be sure that their edge to  $b$  is an  $r$ -edge since  $b$  has a new matching partner  $M(b)$ , so  $b$  must be asked again whether it prefers the new partner over  $a$  when searching for the new  $r$ -edge of  $a$ . The algorithm then repeats Step 2 but starts the search for the new rotation edge of an agent  $a \in A$  at either the previous rotation edge (if  $a \in A \setminus R$ ) or at the direct successor of the new  $M(a)$  in  $\prec_a$  (if  $a \in A \cap R$ ).
4. When a state is reached where it is known for every  $a \in A$  what its  $r$ -edge is (or that it has no  $r$ -edge) but the  $r$ -edges do not form a rotation, the algorithm terminates and outputs  $M$ .

► **Theorem 8.** *Given a stable matching instance with one-sided uncertainty, the algorithm is  $\mathcal{O}(n)$ -competitive for finding a  $B$ -optimal stable matching using comparison queries.*

**Proof.** Let  $\text{OPT}$  denote the number of queries made by an optimal algorithm. Since finding any stable matching can never require more queries than finding a  $B$ -optimal stable matching, Theorem 6 implies that the algorithm makes at most  $\text{OPT}$  queries in the first step.

We analyze the queries executed *after* the first algorithm step. Call a query *good* if it is the first query involving a specific combination of an agent  $a \in A$  and an agent  $p(a) \in B$ , i.e., the first query of form  $\text{prefer}(p(a), a, M(p(a)))$  for that specific combination of  $p(a)$  and  $a$ . All other queries are *bad*. By definition of good queries, the algorithm makes at most  $n^2$  such queries since this is the maximum number of good queries that can exist. Since  $\text{OPT} \geq n - 1$  (Lemma 7), the number of good queries is  $\mathcal{O}(n) \cdot \text{OPT}$ .

Consider the bad queries and a fixed  $a \in A$ . In the second step of the algorithm, it repeatedly executes queries of the form  $\text{prefer}(p(a), a, M(p(a)))$  with  $p(a) \in B$  to find out if  $(a, p(a))$  is an  $r$ -edge, starting with the direct successor  $p(a)$  of  $M(a)$  in  $\prec_a$ . If  $(a, p(a))$  is not an  $r$ -edge, then the next query partner  $p(a)$  for  $a$  moves one spot down in the list  $\prec_a$ . This is repeated until the  $r$ -edge  $(a, r(a))$  of  $a$  is found or we know that  $a$  does not have an  $r$ -edge. Here,  $r(a)$  refers to the element that forms an  $r$ -edge with  $a$ .

If  $a$  does not have an  $r$ -edge, there will be no more queries for  $a$  again as all  $b \in B$  that are lower than  $M(a)$  in the preference list of  $a$  prefer their current partner  $M(b)$  over  $a$  and this partner will only improve during the execution of the algorithm. Otherwise,  $a$  will be considered again in the second step of the algorithm only if a rotation was found in the third step. If  $a$  is part of the rotation, then  $r(a) = p(a)$  will be the new matching partner of  $a$  and  $p(a)$  will be moved one spot down in  $\prec_a$ . Only if  $a$  is not part of the rotation,  $p(a) = r(a)$  remains unchanged by definition of the third step. In conclusion, the next query partner  $p(a)$  of  $a$  moves down one spot in  $\prec_a$  after each query for  $a$  unless a rotation is found that does not contain  $a$ . This means that a bad query for  $a$  can only occur as the first query for  $a$  after a new rotation that does not involve  $a$  is found. Thus, each rotation can cause at most  $|A| - 2$  bad queries (at least two members of  $A$  must be involved in the rotation). Thus, the number of bad queries is at most  $(n - 2) \cdot n_r$  for the number of applied rotations  $n_r$ .

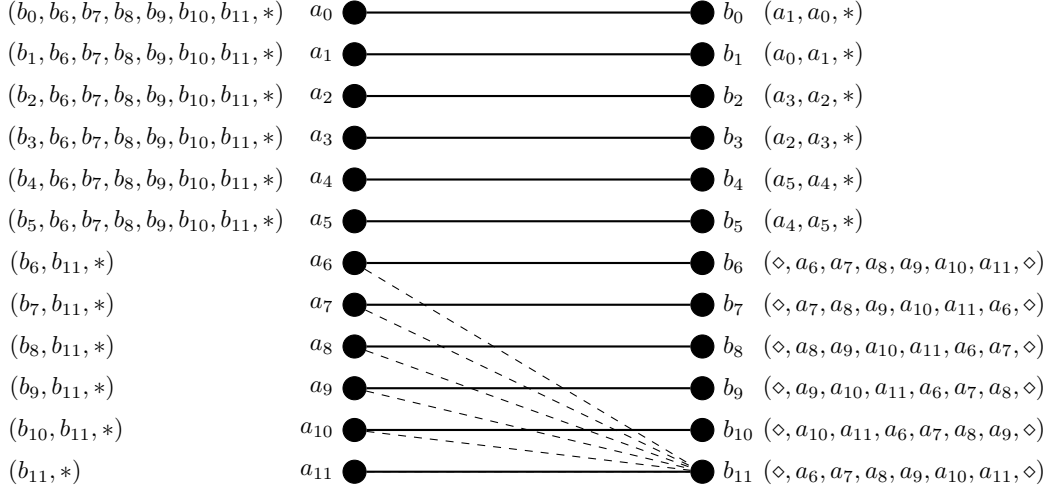
For each applied rotation, at least two agents of  $A$  get re-matched to agents of  $B$  that are lower down on their preference lists than their previous matching partner. This increases the lower bound on the optimal number of queries to show stability (cf. Corollary 5) by at least 2. Thus, Corollary 5 implies  $\text{OPT} \geq 2 \cdot n_r$ . We can conclude that the number of bad queries is at most  $(n - 2) \cdot n_r \leq \mathcal{O}(n) \cdot \text{OPT}$ . ◀

### 3.3.2 Lower Bound for Computing a $B$ -Optimal Matching

We give a lower bound of  $\Omega(n)$  on the competitive ratio for finding a  $B$ -optimal stable matching with comparison queries. This implies that the result of Theorem 8 is, asymptotically, best-possible. Further, the lower bound also holds for verifying that a given matching is  $B$ -optimal.

► **Theorem 9.** *In the comparison query model, every deterministic or randomized online algorithm for finding a  $B$ -optimal stable matching in a stable matching instance with one-sided uncertainty has competitive ratio  $\Omega(n)$ .*

**Proof.** We show the statement for deterministic algorithms and refer to the full version [3] for the extension to randomized ones. The proof of the randomized lower bound exploits Yao's principle [5, 32] and uses a randomized version of the instance we describe here to show the statement. Consider the following instance (cf. Fig. 1) with two sets of agents  $A = \{a_0, \dots, a_{n-1}\}$  and  $B = \{b_0, \dots, b_{n-1}\}$ , and assume  $n/2$  to be even. If this is not the case, then the constant factor in the lower bound will be slightly worse.



■ **Figure 1** Example of the lower bound construction for finding  $B$ -optimal matchings. The solid edges represent the  $A$ -optimal matching  $M$  that needs to be shown to be also  $B$ -optimal using queries. The dashed edges represent rotation edges. Each of the agents in  $\{a_0, a_1, \dots, a_5\}$  also has a rotation edge to some agent in  $\{b_6, b_7, b_8, b_9, b_{10}, b_{11}\}$  that is not shown. An asterisk ( $*$ ) indicates that the remaining agents are placed in arbitrary order in the preference list. A diamond ( $\diamond$ ) indicates that the adversary decides in response to the queries made by the algorithm which of the agents in  $\{a_0, a_1, \dots, a_5\}$  are placed at the front of the preference list and which at the back.

We partition  $A$  into three subsets  $A_1 = \{a_0, \dots, a_{\frac{n}{2}-1}\}$ ,  $A_2 = \{a_{\frac{n}{2}}, \dots, a_{n-2}\}$  and  $A_3 = \{a_{n-1}\}$ , and  $B$  into two subsets,  $B_1 = \{b_0, \dots, b_{\frac{n}{2}-1}\}$  and  $B_2 = B \setminus B_1$ .

In the following, we first define the known  $A$ -side preferences and the adversarial strategy. Then we give bounds on the optimal number of queries and the number of queries made by any deterministic algorithm.

**A-side preferences.** Consider the following preference lists for  $A$ . For an agent  $a_i \in A_1$ , the preference list consists of three parts,  $P(a_i) = P_1(a_i)P_2(a_i)P_3(a_i)$ . The first part of the list is the corresponding  $i^{\text{th}}$  agent of  $B$ , i.e.,  $P_1(a_i) = (b_i)$ . The second part consists of the  $\frac{n}{2}$  agents of set  $B_2$  in increasing order, i.e.,  $P_2(a_i) = (b_{\frac{n}{2}}, b_{\frac{n}{2}+1}, \dots, b_{n-2}, b_{n-1})$ . The last part  $P_3(a_i)$  consists of the agents of  $B_1 \setminus \{b_i\}$  in an arbitrary order.

For an agent  $a_i \in A_2$ , the preference list starts with agent  $b_i$ , followed by the last agent  $b_{n-1}$  and finally an arbitrary order of the remaining agents in group  $B$ . For the single agent  $a_{n-1}$  in set  $A_3$ , the preference list starts with agent  $b_{n-1}$  followed by an arbitrary order of the remaining agents of set  $B$ .

**Adversarial strategy.** The preference lists of the agents in set  $B$  are unknown. The instance has the  $A$ -optimal matching  $M = \{(a_i, b_i) \mid 0 \leq i < n\}$ . The adversary will ensure that this matching is also  $B$ -optimal. Since each  $a_i$  is matched with its top choice, proving stability does not require any queries. To prove  $B$ -optimality of  $M$ , the executed queries must prove that there is no rotation.

The adversary will ensure that  $M$  can be shown to be a  $B$ -optimal matching with  $\mathcal{O}(n)$  queries while any deterministic algorithm is forced to make  $\Omega(n^2)$  queries.

To achieve this, the adversary sets the preferences of the agents of  $B_1$  independent of the algorithm's actions as follows. For each odd  $i \in \{1, 3, 5, \dots, (n/2) - 1\}$ , we let the preference list of  $b_i$  start with  $a_{i-1}$  followed by  $a_i$  and finally all remaining agents in  $A$  in an arbitrary order. The preference list of  $b_{i-1}$  starts with  $a_i$  followed by  $a_{i-1}$  and then the remaining

agents in  $A$  in an arbitrary order. Using these preferences, the sequences  $(a_{i-1}, b_{i-1}), (a_i, b_i)$  are potential rotations. To prove that such a sequence is not a rotation, an algorithm has to show that either  $(a_{i-1}, b_i)$  or  $(a_i, b_{i-1})$  is not an  $r$ -edge. The only way of showing this is to prove that either  $a_{i-1}$  or  $a_i$  instead has an  $r$ -edge to some agent of  $B_2$ .

Consider any deterministic algorithm. The adversary selects the preferences of the agents in  $B_2$  in such a way that the following properties hold:

- (P1) Agent  $a_{n-1}$  has no  $r$ -edge. Note that, by the definition of the preferences of  $B_1$  above,  $a_{n-1}$  already cannot have a rotation edge to an agent of  $B_1$ .
- (P2) Each agent in  $A_2$  has an  $r$ -edge to  $b_{n-1}$ .
- (P3) Each agent  $a_i$  in  $A_1$  has an  $r$ -edge to some agent  $b_{t(i)}$  of  $B_2$ . The choice of that agent  $b_{t(i)}$  depends on the queries made by the algorithm.

The properties (P1)–(P3) ensure that there is no rotation, as the alternating path starting at any  $a \in A \setminus \{a_{n-1}\}$  with the  $r$ -edge of that agent ends at  $a_{n-1}$ , which has no  $r$ -edge.

Let  $t(i)$  denote the index of the agent  $b_{t(i)}$  of  $B_2$  to which  $a_i \in A_1$  has an  $r$ -edge. This index is determined by the adversary in response to the queries made by the algorithm. Concretely, the adversary lets  $t(i)$  be the index of the *last agent*  $b_j \in B_2$  for which the algorithm makes a query of the form  $\text{prefer}(b_j, a_i, *)$ , where we use  $\text{prefer}(b_j, a_i, *)$  as a shorthand to refer to queries  $\text{prefer}(b_j, a_i, a_{i'})$  or  $\text{prefer}(b_j, a_{i'}, a_i)$  for some  $i'$ . If the algorithm doesn't make queries of this form for all  $b_j \in B_2$ , then let  $t(i)$  be an arbitrary  $j$  such that the algorithm does not make a query of this form for  $b_j \in B_2$ . The adversary sets the preferences of the agents in  $B_2$  in such a way that  $b_{t(i)}$  prefers  $a_i$  to her partner  $a_{t(i)}$  in the A-optimal matching  $M$  while all other  $b_j \in B_2$  prefer their partner in the A-optimal matching  $a_j$  to  $a_i$ . For example, if the algorithm was to make queries  $\text{prefer}(b_j, a_i, a_j)$  for all  $b_j \in B_2$  (which it might do in order to check whether  $a_i$  has an  $r$ -edge to one of these agents), the adversary would answer **false** to the first  $\frac{n}{2} - 1$  such queries and **true** to the final one.

To achieve the properties (P1)–(P3), the adversary sets the preferences of each agent  $b_j$  of  $B_2$  as follows:

- $b_j$  prefers  $a_i \in A_1$  to  $a_j$  if and only if  $j = t(i)$ .
- If  $j \neq n - 1$ ,  $b_j$  prefers  $a_j$  to  $a_{j'}$  for all  $j' \neq j, a_{j'} \in A_2$ .
- If  $j = n - 1$ ,  $b_j$  prefers  $a_{j'}$  to  $a_j$  for all  $j' \neq j, a_{j'} \in A_2$ .

This can be done by letting the preference list of  $b_j$  contain first the agents  $a_i \in A_1$  with  $j = t(i)$  in some order, then the agents of  $A_2$  in some order (only ensuring for  $b_j$  that  $a_j$  comes first among the agents of  $A_2$  if  $j \neq n - 1$  and that  $a_j$  comes last among the agents of  $A_2$  if  $j = n - 1$ ), and finally the agents  $a_i \in A_1$  with  $j \neq t(i)$  in some order.

**Upper bound on the optimal query cost.** An optimal solution for the instance can prove that matching  $M$  is  $B$ -optimal by verifying that the properties (P1)–(P3) indeed hold by using at most  $n - 1 + \frac{n}{2} - 1 + \frac{n}{2} = 2n - 2$  queries as follows:

- The  $n - 1$  queries  $\text{prefer}(b_i, a_{n-1}, a_i) = \text{false}$  for  $i \leq n - 2$  show that  $a_{n-1}$  has no  $r$ -edge.
- Each of the  $\frac{n}{2} - 1$  queries  $\text{prefer}(b_{n-1}, a_{\frac{n}{2}+i}, a_{n-1}) = \text{true}$  for  $0 \leq i \leq \frac{n}{2} - 2$  shows that  $a_{\frac{n}{2}+i}$  has an  $r$ -edge to  $b_{n-1}$ . This is because each agent of  $A_2$  has  $b_{n-1}$  in its preference list directly after its current matching partner. So if  $b_{n-1}$  prefers an agent of  $A_2$  over its current partner  $a_{n-1}$ , then this directly gives us an  $r$ -edge.
- Each of the  $\frac{n}{2}$  queries  $\text{prefer}(b_{t(i)}, a_i, a_{t(i)}) = \text{true}$  for  $0 \leq i \leq \frac{n}{2} - 1$  shows that  $a_i$  has a rotation edge to some agent in  $B_2$ . Based on the result of such a query,  $a_i$  must have an  $r$ -edge to either  $b_{t(i)}$  or to some other agent of  $B_2$  that is higher up in  $a_i$ 's preference list.

**Lower bound on the algorithm's query cost.** We provide to the algorithm the information that  $a_{n-1}$  has no  $r$ -edge, that each agent of  $A_2$  has an  $r$ -edge to  $b_{n-1}$ , and we reveal the full preference lists of all agents in  $B_1$ . Clearly, this extra information can only reduce the number of queries a deterministic algorithm may need as it could simply ignore the information.

For each agent  $a_i \in A_1$ , the algorithm will either make queries of the form  $prefer(b_j, a_i, *)$  for all  $b_j \in B_2$  or not. Call  $a_i$  *resolved* in the former case and *unresolved* otherwise. For any resolved agent, the algorithm may have determined that it has an  $r$ -edge to an agent of  $B_2$  and hence cannot be part of a rotation. For the unresolved agents, the algorithm cannot know whether they have an  $r$ -edge to an agent in  $B_2$ .

As argued above, for each odd  $i \in \{1, 3, 5, \dots, \frac{n}{2} - 1\}$ , the algorithm has to resolve either  $a_i$  or  $a_{i-1}$  to prove that  $(a_i, b_i), (a_{i-1}, b_{i-1})$  is not a rotation. Thus, it must resolve at least  $n/4$  agents. For each resolved agent, the algorithm has made queries of the form  $prefer(b_j, a_i, *)$  for each  $b_j \in B_2$ . This totals to at least  $\frac{n}{4} \cdot \frac{n}{2} \cdot \frac{1}{2} = \frac{n^2}{16} \in \Omega(n^2)$  queries. Note that we divide  $\frac{n}{4} \cdot \frac{n}{2}$  by two as a single query  $prefer(b_j, a_i, a_{i'})$  is of the form  $prefer(b_j, a_i, *)$  and also  $prefer(b_j, a_{i'}, *)$ . ◀

### 3.3.3 Offline Results for Computing $B$ -Optimal Stable Matchings

We show NP-hardness for the offline problem of verifying a given matching  $M$  to be stable and  $B$ -optimal. Recall that in the offline problem we assume full knowledge of the  $B$ -side preferences but still want to compute a query set of minimum size that a third party without knowledge of the  $B$ -side preferences could use to verify the  $B$ -optimality of  $M$ .

► **Theorem 10.** *The offline problem of computing an optimal set of comparison queries for finding (or verifying) the  $B$ -optimal stable matching in a stable matching instance with one-sided uncertainty is NP-hard.*

**Proof.** We give a reduction from the NP-hard *Minimum Feedback Arc Set (FAS)* problem. Given a directed graph  $G = (V, E)$ , a feedback arc set is a subset of edges  $E' \subseteq E$  which, if removed from  $G$ , leaves the remaining graph acyclic. The FAS problem is to decide for a given directed graph and some  $k \in \mathbb{Z}_+$ , whether there is a feedback arc set  $E'$  with  $|E'| \leq k$ .

Given an instance of FAS with  $G = (V, E)$  and some  $k$ , we construct a stable matching instance with one-sided uncertainty as follows. For each node  $v$  of  $G$ , introduce an agent  $v$  in  $A$  and an agent  $v'$  in  $B$ . Let  $N^+(v)$  denote the set of out-neighbors of  $v$  in  $G$ , and  $d^+(v) = |N^+(v)|$ . The preference list of  $v$  is such that it ends with  $v'$  followed by all  $u'$  for  $u \in N^+(v)$ . All other  $w'$  in  $B$  come before  $v'$ . Thus, the elements of  $B \setminus \{u' \mid u \in N^+(v)\}$  are the most preferred partners of  $v$ , followed by  $v'$  and finally the elements of  $\{u' \mid u \in N^+(v)\}$ . Let  $M$  be the matching that matches  $v$  to  $v'$ , for all  $v$ . The preference lists of  $b \in B$  are such that  $M$  is the  $B$ -optimal stable matching: Every  $v'$  has  $v$  as top preference, and the remaining agents of  $A$  follow in arbitrary order. By selecting the matching  $M$  this way, we have that, for every  $v \in A$ , all edges to elements of  $\{u' \mid u \in N^+(v)\}$  are potential  $r$ -edges. To prove that such an edge  $(v, u')$  is not an  $r$ -edge, an algorithm has to compare  $u$  and  $v$  from the perspective of  $u'$  to prove that  $u'$  prefers  $M(u') = u$  over  $v$ .

The number of queries  $Q(M)$  needed to verify the stability of  $M$  is determined by  $M$  and is polynomial-time computable by using Theorem 4. To prove  $B$ -optimality of  $M$ , we need to show that there is no rotation (Lemma 1). Indeed, there is a query strategy with  $k$  queries for verifying that there is no rotation if and only if there is a feedback arc set in  $G$  of size  $k$ . To see this, observe that every directed cycle in  $G$  corresponds to a potential rotation in the matching instance, and every query that excludes one of the edges of the potential rotation from being an  $r$ -edge corresponds to the removal of the corresponding arc in  $G$ .

Note that, for the constructed instance, all queries to verify the stability of  $M$  obtain information of the form  $M(b) \prec_b a$  for  $a \in A$  and  $b \in B$  with  $b \prec_a M(a)$ . On the other hand, all queries that help to verify the absence of a rotation obtain information of the form  $M(b) \prec_b a$  for  $a \in A$  and  $b \in B$  with  $M(a) \prec_a b$ . As these are disjoint query sets, we can conclude that there is a query strategy that proves  $M$  to be stable and  $B$ -optimal with at most  $Q(M) + k$  queries if and only if there is a feedback arc set in  $G$  of size at most  $k$ . ◀

We also prove the following approximation for the offline problem by exploiting an  $\mathcal{O}(\log n \log \log n)$ -approximation for weighted feedback arc set by Even et al. [12].

► **Theorem 11.** *The offline problem of computing an optimal set of comparison queries for finding the  $B$ -optimal stable matching in a stable matching instance with one-sided uncertainty can be approximated within ratio  $\mathcal{O}(\log n \log \log n)$ .*

**Proof.** Let  $M$  be the  $B$ -optimal matching. We give an algorithm that verifies  $M$  to be stable and  $B$ -optimal by executing at most  $\mathcal{O}(\log n \log \log n) \cdot \text{OPT}$  queries, where  $\text{OPT}$  is the optimal number of queries for the same instance. First, the algorithm proves that  $M$  is stable using Theorem 4. This leads to at most  $\text{OPT}$  queries.

After that, the algorithm has to prove  $B$ -optimality. First, for every  $a \in A$  that has an  $r$ -edge to an agent  $r(a) \in B$ , the algorithm queries  $\text{prefer}(r(a), a, M(r(a)))$ . Since  $(a, r(a))$  is an  $r$ -edge, this query must return that  $r(a)$  prefers  $a$  over  $M(r(a))$ . This leads to at most  $n \leq \text{OPT} + 1$  queries ( $n \leq \text{OPT} + 1$  holds by Lemma 7). Note that, for an  $a \in A$  with an  $r$ -edge, the query  $\text{prefer}(r(a), a, M(r(a)))$  proves that  $a$  has an  $r$ -edge but is not necessarily sufficient to prove that  $(a, r(a))$  is indeed the  $r$ -edge of  $a$ . If there is an agent  $b \in B$  with  $M(a) \prec_a b \prec_a r(a)$  for which we have not yet verified whether  $b$  prefers  $a$  over  $M(b)$ , then  $(a, b)$  could also still be the  $r$ -edge of  $a$ . We call such pairs  $(a, b)$  *potential  $r$ -edges* and let  $P$  denote the set of these edges.

It remains to consider the graph  $G$  defined by the matching edges, the  $r$ -edges  $R$ , and all potential  $r$ -edges  $P$ . If  $G$  has no cycle alternating between edges in  $M$  and edges in  $P \cup R$ , then we have shown that  $M$  does not expose a rotation and, thus, is  $B$ -optimal. Otherwise, the algorithm has to execute queries  $\text{prefer}(b, a, M(b))$  for edges  $(a, b) \in P$  to prove that they are not actually  $r$ -edges until it becomes clear that  $M$  has no rotation.

To select the edges  $(a, b) \in P$  for which the algorithm executes such queries, we exploit the  $\mathcal{O}(\log n \log \log n)$ -approximation for weighted feedback arc set by Even et al. [12]. To this end, we create an instance of the weighted feedback arc set problem by considering the vertices  $A \cup B$ , adding the edges  $M \cup R$  with weight  $\infty$  each and adding the edges  $P$  with weight 1 each. We orient all edges in  $M$  from the  $B$ -side vertex to the  $A$ -side vertex and all edges in  $R \cup P$  from the  $A$ -side vertex to the  $B$ -side vertex. The orientation ensures that all cycles in the graph alternate between  $M$ -edges and  $R \cup P$ -edges. Since the matching  $M$  is  $B$ -optimal by assumption, there cannot be an alternating cycle using only edges in  $M \cup R$ , so there must be a feedback arc set that only uses edges in  $P$ . The choice of the edge weights ensures that every approximation algorithm for weighted feedback arc set finds such a solution. We use the  $\mathcal{O}(\log n \log \log n)$ -approximation to find such a feedback arc set  $F \subseteq P$ . Since removing  $F$  from the instance yields an acyclic graph, querying  $\text{prefer}(b, a, M(b))$  for each  $(a, b) \in F$  proves that  $M$  does not expose a rotation. As the minimum weight feedback arc set is the cheapest way to prove that  $M$  does not have a rotation, we have  $|F| \leq \mathcal{O}(\log n \log \log n) \cdot \text{OPT}$ , which implies the theorem. ◀

## 4 Stable Matching with Interview Queries

In this section, we summarize our results for the interview query model. We refer to the full version [3] for formal proofs. Most of our results and proofs are quite similar to their counterparts for comparison queries. This might be surprising as interview and comparison queries are, in a sense, incomparable: While interview queries allow us to more efficiently determine full preference lists, a comparison between two agents can be done more efficiently via a single comparison query. As we show the same (asymptotic) bounds on the competitive ratio, the latter seems to be the deciding factor.

► **Theorem 12.** *In the interview query model, the best possible (randomized) competitive ratio for finding the  $B$ -optimal stable matching in an instance of stable matching with one-sided uncertainty is in  $\Theta(n)$ .*

For the offline problem of verifying a given  $B$ -optimal stable matching with interview queries, Rastegari et al. [30] show NP-hardness in a setting with partial uncertainty on both sides. As their proof exploits the possibility of giving partial information as part of the input, it does not directly translate to our setting with one-sided uncertainty.

► **Theorem 13.** *The offline problem of computing an optimal set of interview queries for finding the  $B$ -optimal stable matching in a stable matching instance with one-sided uncertainty is NP-hard.*

A 1-competitive algorithm for finding a stable matching and verifying a given stable matching with interview queries is implied by the results and arguments from [30] for a more general uncertainty setting.

## 5 Stable Matching with Set Queries

We consider the stable matching problem with one-sided uncertainty and set queries. Note that set queries are a natural generalization of comparison queries. For verifying any  $B$ -optimal matching, we show that the optimal number of set queries is at least  $n - 1$ . We also observe that there is an algorithm that makes at most  $n^2$  queries for finding the  $B$ -optimal matching (or an  $A$ -optimal matching if we want to), as one can sort all preference lists using  $n^2$  set queries. This implies an  $\mathcal{O}(n)$ -competitive algorithm for finding the  $B$ -optimal matching. For the subproblem of verifying that a given matching is  $B$ -optimal, we give an  $\mathcal{O}(\log n)$ -competitive algorithm by exploiting the additional power of set queries in an involved binary search algorithm. If we only have to verify stability for a given matching, we give a 1-competitive algorithm. Furthermore, we show that the offline problem of verifying that a given matching does not have a rotation is NP-hard.

### 5.1 Verifying That a Given Matching Is Stable

We start by characterizing the optimal number of queries (and query strategy) to verify that a given matching  $M$  is stable. The main difference to the comparison model is that, for a fixed  $b \in B$ , a single query  $\text{top}(b, \{a \mid b \prec_a M(a)\} \cup \{M(b)\})$  is sufficient to prove that  $b$  is not part of any blocking pair.

► **Theorem 14.** *Consider a stable matching instance with one-sided uncertainty and a stable matching  $M$ . The minimum number of set queries to verify that  $M$  is stable is  $|\{b \in B \mid \exists a \in A: b \prec_a M(a)\}| \leq n$ . Further, there is a 1-competitive algorithm to verify that  $M$  is stable.*

**Proof.** Consider an arbitrary  $b \in B$ . Let  $Z(b) = \{a \in A \mid b \prec_a M(a)\}$ , i.e.,  $Z(b)$  contains all  $a \in A$  that could potentially form a blocking pair with  $b$ . Thus,  $M$  can only be stable if  $M(b) \prec_b a$  holds for all  $b \in B$  and  $a \in Z(b)$ . If  $Z(b) \neq \emptyset$ , at least one query to  $b$  is necessary, and the query  $top(b, Z(b) \cup \{M(b)\})$  with answer  $M(b)$  reveals all the required information to prove that  $b$  is not part of any blocking pair. Thus, the minimum number of queries to confirm that  $M$  is stable is  $|\{b \in B \mid \exists a \in A: b \prec_a M(a)\}|$  as claimed. Furthermore, the algorithm that queries  $top(b, Z(b) \cup \{M(b)\})$  for all  $b \in B$  with  $Z(b) \neq \emptyset$  is 1-competitive.  $\blacktriangleleft$

## 5.2 Verifying That a Given Matching Is Stable and $B$ -Optimal

For the problem of confirming that a given matching is  $B$ -optimal by using set queries, we show that every algorithm needs to execute at least  $n - 1$  queries. This is analogous to the setting with comparison queries and uses a similar proof as Lemma 7. It implies that finding a  $B$ -optimal matching also requires at least  $n - 1$  queries.

► **Lemma 15.** *Consider an arbitrary stable matching instance with one-sided uncertainty and the  $B$ -optimal matching  $M$ . Every algorithm needs at least  $n - 1$  set queries to verify that  $M$  is indeed stable and  $B$ -optimal.*

**Proof.** For each  $b \in B$ , let  $Z(b) = \{a \in A \mid b \prec_a M(a)\}$  and let  $S = \{b \in B \mid Z(b) \neq \emptyset\}$ . By the proof of Theorem 14, every algorithm needs to execute at least one query of the form  $top(b, X)$  with  $X \subseteq A$  for all  $b \in S$  and this query has to return  $M(b)$  as the top choice. Since verifying  $B$ -optimality includes proving stability, this leads to at least  $|S|$  queries.

Consider an arbitrary algorithm that verifies  $M$  to be  $B$ -optimal and let  $A_1 \subseteq A$  denote the agents of  $A$  that are returned as the top choice by some query of the algorithm. Then  $|S| \leq |A_1|$  and  $\{a \in A \mid \exists b \in S: M(b) = a\} \subseteq A_1$  by the argumentation above.

If  $|A_1| \geq n - 1$ , then the statement follows immediately, so assume  $|A_1| < n - 1$  and let  $A_2 = A \setminus A_1$ . Since  $|A_1| < n - 1$ , the set  $A_2$  has at least two distinct members  $a_1$  and  $a_2$ . Furthermore, we must have  $M(a_1), M(a_2) \notin S$  as observed above. By definition of  $S$ , we have  $M(a_1) \prec_{a_1} M(a_2)$  and  $M(a_2) \prec_{a_2} M(a_1)$ . This means that  $(a_1, M(a_2))$  and  $(a_2, M(a_1))$ , based on the initially given information, could potentially be rotation edges. Thus,  $(a_1, M(a_1)), (a_2, M(a_2))$  could potentially be a rotation and the algorithm has to prove that this is not the case by showing that one of  $(a_1, M(a_2))$  and  $(a_2, M(a_1))$  is not an  $r$ -edge. To prove that  $(a_1, M(a_2))$  is not an  $r$ -edge, one has to either verify  $a_2 \prec_{M(a_2)} a_1$  or  $a_1 \prec_b M(b)$  for some  $b \in B$  with  $M(a_1) \prec_{a_1} b \prec_{a_1} M(a_2)$ . However, this requires at least one query that returns either  $a_1$  or  $a_2$  as the top choice, and there is a symmetric argument for proving that  $(a_2, M(a_1))$  is not an  $r$ -edge. Since  $a_1$  and  $a_2$  are never returned as the top choice by a query of the algorithm, this is a contradiction to the assumption that the algorithm verifies that  $M$  is  $B$ -optimal.  $\blacktriangleleft$

In contrast to the comparison model, there exists an offline algorithm that asymptotically matches the lower bound of Lemma 15.

► **Theorem 16.** *There exists a polynomial-time offline algorithm that, given an instance of stable matching with one-sided uncertainty and the  $B$ -optimal matching  $M$ , verifies that  $M$  is indeed stable and  $B$ -optimal by executing  $\mathcal{O}(n)$  set queries.*

**Proof.** By the proof of Theorem 14, an algorithm can prove  $M$  to be stable by executing at most  $n$  set queries, so it remains to prove that  $M$  is  $B$ -optimal by executing at most  $\mathcal{O}(n)$  set queries.



We do so by proving that  $M$  does not contain a rotation. First, for each  $b \in B$ , we compute the set  $P(b) = \{a \in A \mid M(a) \prec_a b \text{ and } M(b) \prec_b a\}$ . Each tuple  $(b, a)$  with  $b \in B$  and  $a \in P(b)$  could be a rotation edge based on  $\prec_a$  but is not a rotation edge as  $M(b) \prec_b a$ . An algorithm can prove that none of these edge are actually rotation edges by executing a query  $\text{top}(b, P(b) \cup \{M(b)\})$  for each  $b \in B$ . This leads to  $n$  additional queries.

If an  $a \in A$  does not have a rotation edge, then the previous queries prove that this is the case. Consider an  $a \in A$  that has a rotation edge. Then the second endpoint of that edge is the agent  $b \in B$  of highest preference according to  $\prec_a$  among those agents that satisfy  $M(a) \prec_a b$  and  $a \prec_b M(b)$ . Let  $b$  be that endpoint. To prove that  $(a, b)$  is indeed a rotation edge, an algorithm has to verify  $a \prec_b M(b)$  and  $M(b') \prec_{b'} a$  for all  $b'$  with  $M(a) \prec_a b' \prec_a b$ . The latter has already been verified by the previous  $n$  queries and the former can be proven by an additional query  $\text{top}(b, \{a, M(b)\})$ . Doing this for every  $a \in A$  that has a rotation edge leads to at most  $n$  further queries.

Executing these queries yields, for each  $a \in A$ , either the rotation edge of  $a$  or a proof that  $a$  does not have a rotation edge. Thus, it gives sufficient information to show that  $M$  does not have a rotation and is  $B$ -optimal.  $\blacktriangleleft$

Next, we give an online algorithm that decides whether a given matching  $M$  is  $B$ -optimal by executing at most  $\mathcal{O}(n \log n)$  set queries. In combination with Lemma 15, this yields an  $\mathcal{O}(\log n)$ -competitive algorithm for verifying that a given matching is  $B$ -optimal with set queries.

► **Theorem 17.** *There is an algorithm that decides if a given matching  $M$  in a stable matching instance with one-sided uncertainty is stable and  $B$ -optimal with  $\mathcal{O}(n \log n)$  set queries.*

**Proof.** First, we can use Theorem 14 and execute  $\mathcal{O}(n)$  queries to decide whether  $M$  is stable. If  $M$  turns out not to be stable, then we are done. Otherwise, we have to decide whether  $M$  is  $B$ -optimal by using at most  $\mathcal{O}(n \log n)$  set queries. We do so by giving an algorithm that, for each  $a \in A$ , either finds the rotation edge of  $a$  or proves that  $a$  does not have a rotation edge. After executing that algorithm we clearly have sufficient information to decide whether  $M$  exposes a rotation and, thus, whether it is  $B$ -optimal.

For each  $a \in A$ , we use  $R(a)$  to refer to the set of agents that could potentially form a rotation edge with  $a$ . Initially, we set  $R(a) = \{b \in B \mid M(a) \prec_a b\}$  as all agents with a lower priority than  $M(a)$  can potentially form a rotation edge with  $a$  based on the initially given information. During the course of our algorithm, we will update the set  $R(a)$  such that it always only contains the agents of  $B$  that, based on the information obtained by all previous queries, could still form a rotation edge with  $a$ . In particular, if we obtain the information that  $M(b) \prec_b a$  for some  $b \in R(a)$ , then  $(a, b)$  clearly cannot be a rotation edge and we can update  $R(a) = R(a) \setminus \{b\}$ . Similarly, if we obtain the information that  $a \prec_b M(b)$  for some  $b \in R(a)$ , then the agents  $b' \in R(a)$  with  $b \prec_a b'$  cannot form a rotation edge with  $a$  anymore and we can update  $R(a) = R(a) \setminus \{b' \in R(a) \mid b \prec_a b'\}$ . Given the current list  $R(a)$  of potential rotation edge partners, we use  $\bar{R}(a)$  to refer to the  $\left\lceil \frac{|R(a)|}{2} \right\rceil$  agents of  $R(a)$  with the highest priority in  $R(a)$  according to  $\prec_a$ .

Our algorithm, cf. Algorithm 2, proceeds in iterations that each execute at most  $\mathcal{O}(n)$  set queries. Let  $R_i(a)$ ,  $a \in A$ , denote the current sets of potential rotation edges at the beginning of iteration  $i$  and let  $\bar{R}_i(a)$  be as defined above. We define our algorithm in a way such that each iteration  $i$  decides for each  $a \in A$  whether it has a rotation edge to an agent of  $\bar{R}_i(a)$  or not. Then,  $|R_{i+1}(a)| \leq \frac{|R_i(a)|+1}{2}$  holds for each  $a \in A$  with  $|R_i(a)| > 1$  as we either get  $R_{i+1}(a) \subseteq \bar{R}_i(a)$  or  $R_{i+1}(a) \subseteq R_i(a) \setminus \bar{R}_i(a)$ . Furthermore, if  $|R_i(a)| = 1$ , then iteration  $i$  either identifies the rotation edge of  $a$  or proves that it does not have one. This means that

■ **Algorithm 2** Algorithm to decide whether a given matching is  $B$ -optimal using set queries.

---

**Input:** Stable matching instance with one-sided uncertainty and a matching  $M$ .

- 1 Decide whether  $M$  is stable using Theorem 14. If  $M$  is not stable, terminate;
- 2  $R(a) \leftarrow \{b \in B \mid M(a) \prec_a b\}$  for all  $a \in A$ ;
- 3 **while** *We did not decide yet whether  $M$  is  $B$ -optimal* **do**
- 4      $U \leftarrow \{a \in A \mid |\bar{R}(a)| \geq 1\}$ ;
- 5     **for**  $b \in B$  **do**
- 6          $U_b \leftarrow \{a \in U \mid b \in \bar{R}(a)\}$ ;
- 7         **repeat**
- 8              $t \leftarrow \text{top}(b, U_b \cup \{M(b)\})$ ;
- 9             **if**  $t = M(b)$  **then**  $R(a) \leftarrow R(a) \setminus b$  for all  $a \in U_b$ ;  $U_b \leftarrow \emptyset$  ;
- 10            **else**
- 11                 $U \leftarrow U \setminus \{t\}$ ;  $U_b \leftarrow U_b \setminus \{t\}$ ;
- 12                 $R(t) \leftarrow R(t) \setminus \{b' \in R(t) \mid b \prec_t b'\}$ ;
- 13         **until**  $U_b = \emptyset$ ;

---

after at most  $\mathcal{O}(\log n)$  such iterations, for each  $a \in A$ , we either found the rotation edge of  $a$  or verified that it does not have one. Since each iteration executes  $\mathcal{O}(n)$  set queries, we get an algorithm that executes  $\mathcal{O}(n \log n)$  set queries and decides whether  $M$  is  $B$ -optimal.

It remains to show that each iteration  $i$  indeed executes  $\mathcal{O}(n)$  set queries and decides, for each  $a \in A$ , whether  $a$  has a rotation edge to some agent of  $\bar{R}_i(a)$ . Lines 4 to 13 of Algorithm 2 show the pseudocode for such an iteration. In each iteration  $i$ , the algorithm considers the set  $U = \{a \in A \mid |\bar{R}_i(a)| \geq 1\}$ , i.e., the subset of  $A$  for which we do not yet know whether it has a rotation edge to some agent of  $\bar{R}_i(a)$ . Then, the algorithm iterates through the agents  $b$  of  $B$  and considers the set  $U_b = \{a \in U \mid b \in \bar{R}_i(a)\}$ . Note that, for each  $a \in U_b$ , it holds that if  $a \prec_b M(b)$ , then  $a$  has a rotation edge to some agent of  $\bar{R}_i(a)$  (not necessarily to  $b$ ). The algorithm executes the query  $\text{top}(b, U_b \cup \{M(b)\})$ . If this query returns  $M(b)$ , then we know for sure that  $b$  does not have a rotation edge to any agent of  $U_b$  and we can discard  $b$  for the rest of the iteration and also remove  $b$  from the current  $R(a)$  of all  $a \in U_b$ . On the other hand, if the query returns  $a \neq M(b)$ , then we know that  $a$  has a rotation edge to some agent of  $\bar{R}_i(a)$  and we do not need to consider  $a$  for the rest of the iteration anymore. Thus, after each query within the iteration we discard an agent of either  $A$  or  $B$ , which means that the iteration terminates after at most  $2n$  queries. At the end of the iteration, we know for each  $a \in A$  whether it has a rotation edge to some  $b \in \bar{R}_i(a)$ . ◀

For the offline problem, we show that computing the query set of minimum size that verifies that a given matching does not have a rotation is NP-hard. However, in the instances constructed by the reduction, verifying that the given matching does not have a rotation *and is stable* is trivial as we will discuss after the proof. This means that the following result does *not* imply NP-hardness for the offline variant of finding the  $B$ -optimal matching with set queries.

► **Theorem 18.** *In the set query model, the offline problem of computing an optimal set of queries for verifying that a given  $B$ -optimal stable matching  $M$  for a stable matching instance with one-sided uncertainty does not have a rotation is NP-hard.*

**Proof.** We show the statement by reduction from the NP-hard *feedback vertex set problem* [23]. In this problem, we are given a directed graph  $G = (V, E)$  and a parameter  $k \in \mathbb{N}$ . The goal is to decide whether there exists a subset  $F \subseteq V$  with  $|F| \leq k$  such that deleting  $F$  from  $G$  yields an acyclic graph.

We construct an instance of the stable matching problem with one-sided uncertainty and a matching  $M$  as follows:

1. For each  $v \in V$ , we add an agent  $a_v$  to set  $A$  and a matching partner  $M(a_v)$  to set  $B$ .
2. For each  $v \in V$  and  $u \in V \setminus \{v\}$ , we set  $M(a_v) \prec_{a_v} M(a_u)$  if  $(v, u) \in E$  and  $M(a_u) \prec_{a_v} M(a_v)$  otherwise.
3. For each  $v \in V$  and  $u \in V \setminus \{v\}$ , we set  $a_v \prec_{M(v)} a_u$ .

Based on the  $A$ -side preferences, each  $(a_v, M(a_u))$  with  $(v, u) \in E$  could be an  $r$ -edge and each  $(a_v, M(a_u))$  with  $(v, u) \notin E$  is not an  $r$ -edge. Consider the directed graph  $G' = (A \cup B, E')$  with  $E' = \{(M(a_v), a_v) \mid v \in V\} \cup \{(a_v, M(a_u)) \mid (v, u) \in E\}$ . Then, based on the  $A$ -side preferences, each cycle in  $G'$  could be a rotation. Furthermore, if we contract the edges  $\{(M(a_v), a_v) \mid v \in V\}$ , we arrive at the given graph  $G$ .

Assume that there is a set  $F \subseteq V$  with  $|F| \leq k$  such that deleting  $F$  from  $G$  yields an acyclic graph. Consider the queries  $\text{top}(M(a_v), A)$  for all  $v \in F$ . By the third step of the reduction, these queries prove that the agents  $M(a_v)$  with  $v \in F$  are not part of any rotation. This also means that the agents  $a_v$  with  $v \in F$  cannot be part of a rotation. Thus, the only edges that can still be part of a rotation are the matching edges  $(M(a_v), a_v)$  with  $v \notin F$  and the edges  $(a_v, M(a_u))$  with  $(v, u) \in E$  but  $v, u \notin F$ . If we consider the graph induced by these remaining edges and contract the matching edges, we arrive at the subgraph  $G[V \setminus F]$  of the given feedback vertex set instance. Since this graph by assumption does not contain a cycle, this implies that executing the queries proves that the constructed instance has no rotation.

Consider a query strategy that proves the constructed instance to not have a rotation by using at most  $k$  queries. Let  $A' \subseteq A$  denote the set of all agents that are returned as the top choice by at least one of those queries. Then, by construction, the alternative query strategy that queries  $\text{top}(M(a_v), A)$  for each  $a_v \in A'$  must also be feasible and uses at most  $k$  queries. This alternative strategy proves that there exists no rotation by proving that no  $a_v \in A'$  is part of any rotation. Thus, removing all vertices  $a_v$  and  $M(a_v)$  with  $a_v \in A'$  from the graph  $G'$  as defined above yields a graph without cycles. This also implies that removing  $F = \{v \in V \mid a_v \in A'\}$  from  $G$  yields a graph without cycles. Thus,  $F$  with  $|F| \leq k$  is feasible for the given feedback vertex set instance. ◀

In the instances constructed within the proof, querying  $\text{top}(M(a_v), A)$  for between  $n - 1$  and  $n$  agents  $a_v \in A$  proves that the given matching is stable and  $B$ -optimal. If  $n - 1$  queries suffice, then this is optimal by Lemma 15. Otherwise,  $n$  queries are optimal. We can decide whether  $n - 1$  queries suffice via enumeration over which agent  $M(a_v)$  does not receive a query  $\text{top}(M(a_v), A)$ . Thus, the NP-hardness for proving that no rotation exists does not directly translate to the offline problem of proving that a given matching has no rotation and is stable.

## 6 Open Problems

While we understand the comparison model quite rigorously, it remains open in the set query model what best possible competitive ratio can be achieved for finding a ( $A$ - or  $B$ -optimal) stable matching. Further, it would be interesting to investigate the two-sided stable matching problem with uncertainty in the preference lists on both sides. For verifying the stability of a given matching in this case, we give a best possible 2-competitive algorithm in the full version [3]. All other questions regarding finding a stable or stable and optimal matching remain open under two-sided uncertainty. It would also be interesting to investigate a generalized set query model in which a query to a set  $S \subseteq A$  for a  $b \in B$  reveals the top- $k$  partners of  $b$ , that is, the  $k$  partners in  $S$  that  $b$  prefers most.




## References

- 1 Haris Aziz, Péter Biró, Ronald de Haan, and Baharak Rastegari. Pareto optimal allocation under uncertain preferences: uncertainty models, algorithms, and complexity. *Artif. Intell.*, 276:57–78, 2019.
- 2 Haris Aziz, Péter Biró, Serge Gaspers, Ronald de Haan, Nicholas Mattei, and Baharak Rastegari. Stable matching with uncertain linear preferences. *Algorithmica*, 82(5):1410–1433, 2020.
- 3 Evripidis Bampis, Konstantinos Dogeas, Thomas Erlebach, Nicole Megow, Jens Schläöter, and Amitabh Trehan. Competitive query minimization for stable matching with one-sided uncertainty, 2024. [arXiv:2407.10170](https://arxiv.org/abs/2407.10170).
- 4 Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schläöter. Orienting (hyper)graphs under explorable stochastic uncertainty. In *ESA*, volume 204 of *LIPICs*, pages 10:1–10:18, 2021. doi:10.4230/LIPICs.ESA.2021.10.
- 5 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
- 6 Joanna Drummond and Craig Boutilier. Elicitation and approximately stable matching with partial preferences. In *IJCAI*, pages 97–105. IJCAI/AAAI, 2013.
- 7 Joanna Drummond and Craig Boutilier. Preference elicitation and interview minimization in stable matchings. In *AAAI*, pages 645–653. AAAI Press, 2014.
- 8 Christoph Dürr, Thomas Erlebach, Nicole Megow, and Julie Meißner. An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675, 2020.
- 9 Lars Ehlers and Jordi Massó. Matching markets under (in)complete information. *J. Econ. Theory*, 157:295–314, 2015.
- 10 T. Erlebach and M. Hoffmann. Query-competitive algorithms for computing with uncertainty. *Bulletin of the EATCS*, 116:22–39, 2015. URL: <http://bulletin.eatcs.org/index.php/beatcs/article/view/335>.
- 11 Thomas Erlebach, Murilo S. de Lima, Nicole Megow, and Jens Schläöter. Sorting and hypergraph orientation under uncertainty with predictions. In *IJCAI*, pages 5577–5585. ijcai.org, 2023.
- 12 Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.
- 13 D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. doi:10.1080/00029890.1962.11989827.
- 14 Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. *Games Econ. Behav.*, 118:626–647, 2019. doi:10.1016/j.geb.2018.10.013.
- 15 Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem – Structure and Algorithms*. Foundations of computing series. MIT Press, 1989.
- 16 Guillaume Haeringer and Vincent Iehlé. Two-sided matching with one-sided preferences. In *EC*, page 353. ACM, 2014.
- 17 Guillaume Haeringer and Vincent Iehlé. Two-sided matching with (almost) one-sided preferences. *American Economic Journal: Microeconomics*, 11(3):155–190, 2019.
- 18 Guillaume Haeringer and Vincent Iehlé. Enjeux stratégiques du concours de recrutement des enseignants chercheurs. *Revue Economique*, 61(4):697–721, 2010.
- 19 M. M. Halldórsson and M. S. de Lima. Query-competitive sorting with uncertainty. In *MFCS*, volume 138 of *LIPICs*, pages 7:1–7:15, 2019. doi:10.4230/LIPICs.MFCS.2019.7.
- 20 Michael Hoffmann, Thomas Erlebach, Danny Krizanc, Matús Mihalák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In *STACS*, volume 1 of *LIPICs*, pages 277–288. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2008.
- 21 Hadi Hosseini, Vijay Menon, Nisarg Shah, and Sujoy Sikdar. Necessarily optimal one-sided matchings. In *AAAI*, pages 5481–5488. AAAI Press, 2021.
- 22 S. Kahan. A model for data in motion. In *STOC’91: 23rd Annual ACM Symposium on Theory of Computing*, pages 265–277, 1991. doi:10.1145/103418.103449.

- 23 Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming*, pages 219–241. Springer, 2010.
- 24 Thomas Ma, Vijay Menon, and Kate Larson. Improving welfare in one-sided matchings using simple threshold queries. In *IJCAI*, pages 321–327. ijcai.org, 2021.
- 25 David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science*. WorldScientific, 2013. doi:10.1142/8591.
- 26 N. Megow, J. Meißner, and M. Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017. doi:10.1137/16M1088375.
- 27 Cheng Ng and Daniel S. Hirschberg. Lower bounds for the stable marriage problem and its variants. *SIAM J. Comput.*, 19(1):71–77, 1990. doi:10.1137/0219004.
- 28 Jannik Peters. Online elicitation of necessarily optimal matchings. In *AAAI*, pages 5164–5172. AAAI Press, 2022.
- 29 Baharak Rastegari, Anne Condon, Nicole Immorlica, Robert W. Irving, and Kevin Leyton-Brown. Reasoning about optimal stable matchings under partial information. In *EC*, pages 431–448. ACM, 2014.
- 30 Baharak Rastegari, Anne Condon, Nicole Immorlica, and Kevin Leyton-Brown. Two-sided matching with partial information. In *EC*, pages 733–750. ACM, 2013.
- 31 Alvin E. Roth and Marilda A. Oliveira Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Econometric Society Monographs. Cambridge University Press, 1990. doi:10.1017/CCOL052139015X.
- 32 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity. In *FOCS*, pages 222–227. IEEE Computer Society, 1977.



# A Constant Factor Approximation for Directed Feedback Vertex Set in Graphs of Bounded Genus

Hao Sun   

University of Alberta, 116 St & 85 Ave, Edmonton, AB T6G 2R3, Canada

---

## Abstract

The minimum directed feedback vertex set problem consists in finding the minimum set of vertices that should be removed in order to make a directed graph acyclic. This is a well-known NP-hard optimization problem with applications in various fields, such as VLSI chip design, bioinformatics and transaction processing deadlock prevention and node-weighted network design. We show a constant factor approximation for the directed feedback vertex set problem in graphs of bounded genus.

**2012 ACM Subject Classification** Mathematics of computing → Graphs and surfaces; Mathematics of computing → Approximation algorithms; Mathematics of computing → Graph algorithms; Theory of computation → Rounding techniques; Theory of computation → Packing and covering problems

**Keywords and phrases** Feedback Vertex Set, Combinatorial Optimization, Approximation Algorithms, min-max relation, linear programming

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.18

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2311.01026>

**Acknowledgements** The author would like to thank Zachary Friggstad for invaluable guidance in the writing process and Jochen Koenemann for suggesting the problem.

## 1 Introduction

In the directed feedback vertex set problem (DFVS), we are given a (node-weighted) directed graph  $G = (V, E)$  with costs  $c_v \forall v \in V$  and wish to find a minimum cost set  $X$  for which  $G \setminus X$  is acyclic. DFVS is one of Karp's original 21 NP-hard problems [16]. The DFVS problem has many applications including deadlock resolution [10], VLSI chip design [19] and program verification [20].

A 2-approximation for (undirected) FVS is given in [2]. DFVS has a 2-approximation in tournaments [21] and bipartite tournaments [24], is polynomial-time solvable on graphs of bounded treewidth, has a 2.4-approximation in planar graphs [3] and has an  $O(\log n \log \log n)$ -approximation in general graphs [7]. DFVS does not have an  $O(1)$ -approximation under the unique games conjecture [13]. The genus of a graph is the minimal integer  $g$  such that the graph can be drawn without crossing itself on a sphere with  $g$  handles.

The following is the natural LP for DFVS and its dual, where  $\mathcal{C}$  is the set of directed cycles of our graph.

$$\begin{array}{lcl} \min & c^T x & (\text{P}_{\text{DFVS}}) \\ \text{s.t.} & x(C) \geq 1 \quad \forall C \in \mathcal{C} & (1) \\ & x \geq 0 & \end{array} \quad \left| \quad \begin{array}{lcl} \max & \mathbf{1}^T y & (\text{D}_{\text{DFVS}}) \\ \text{s.t.} & \sum_{C \in \mathcal{C}, v \in C} y_C \leq c_v \quad \forall v \in V(G) & \\ & y \geq 0 & . \end{array}$$



© Hao Sun;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 18; pp. 18:1–18:20



Leibniz International Proceedings in Informatics

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

Given that constant approximations for DFVS exist for planar graphs, one naturally wonders if DFVS admits constant approximations in bounded genus graphs. We answer this question positively.

► **Theorem 1.** *For any fixed genus  $g$ , there is a polynomial-time  $O(g)$ -approximation for DFVS for graphs of genus  $g$ . Moreover, the algorithm returns a DFVS with cost  $O(g)$  times the optimum solution to  $(P_{DFVS})$ .*

From the proof of Theorem 1, it is clear that the algorithm in Theorem 1 runs in time  $O(g)\text{poly}(|V(G)|)$ .

For uniform costs, the dual LP  $(D_{DFVS})$  is the natural LP of the *dicycle packing problem*. The dicycle packing problem is the problem of finding the maximum number of vertex disjoint dicycles of a graph. Schlomberg et al. [23] show that the LP gap of the natural LP for dicycle packing is at most  $\Omega(\frac{1}{g^2 \log g})$  on any graph of genus  $g$ . Our result then also implies that the minimum size of a DFVS is at most  $O(g^3 \log g)$  the size of a maximum dicycle packing.

## 1.1 Our techniques

Informally speaking, for a (directed) graph embedded on a surface where each directed cycle bounds a region homeomorphic to the plane, one can apply the same primal-dual techniques in [12, 3] to obtain a constant factor primal-dual approximation.

In the other case, our algorithm will use the natural LP for DFVS to look for a “separator” [5, 8, 9]  $S \subset V$  of cost at most a constant times the optimal DFVS such that  $G \setminus S$  is of smaller genus. We obtain a directed cycle  $C$ , the removal of which results in a surface of one smaller genus. Traversing along the dicycle, we may define a “left” and “right” side of the dicycle. Like in [7], we solve the DFVS LP and use the LP values as distances.

If there is no short path leaving  $C$  from the left and entering  $C$  from the right and vice versa then there is a small separator  $S$  such that each dicycle of  $G \setminus S$  either does not use any “left arc” that is, an arc coming in or leaving  $C$  from the left, or does not use any “right arc” that is, an arc coming in or leaving  $C$  from the right.  $G$  with all left (resp. right) arcs deleted is of genus at least one less so inductively we can solve within a constant factor DFVS on  $G$  with all left (resp. right) arcs deleted. These two solutions together with  $S$  form a DFVS of constant times more than the optimum. If such short paths exist but all starting points of such paths and ending points of such paths are far apart the analysis is similar.

The final case is where there are short paths  $P_1, P_2$  leaving  $C$  from the left and entering  $C$  from the right (or vice versa) and the starting point of  $P_2$  is close to the endpoint of  $P_1$ . We show that  $P_1$  is “far” from  $P_2$  so to speak (we are using directed distances so this is not the same as  $P_2$  being far from  $P_1$ ) and compute a suitable separator. We show that the resulting strongly connected components are of smaller genus. We then combine approximations for different components to give an approximation for the original graph.

The presentation in this paper is focused on demonstrating linear dependence on the genus rather than optimizing the constant in Theorem 1.

## 2 Preliminaries

In our figures, we will use the representation of the torus by taking the unit square  $[0, 1] \times [0, 1]$  and identifying the two pairs of edges  $\{0\} \times [0, 1]$ ,  $\{1\} \times [0, 1]$  and  $[0, 1] \times \{0\}$ ,  $[0, 1] \times \{1\}$ , that is, the point  $(0, p)$  is identified with  $(1, p)$  and the point  $(q, 0)$  is identified with the point  $(q, 1)$ .



Throughout this paper, all surfaces are orientable and smooth and all curves are piecewise smooth. Distances on surfaces will refer to the geodesic (shortest path on the surface) distance. It is well known (see for instance [22]) that a smooth orientable surface  $Q$  is diffeomorphic to the  $g$ -genus torus for some  $g$ . For  $X \subset Q$ , denote  $\text{cl}(X)$  as the closure of  $X$  in  $Q$ . We henceforth assume our surface is a  $g$ -genus torus for some  $g$ .

Let us call a cycle of  $G$ , or a closed curve  $C$  embedded on a surface  $Q$  *facial* if it bounds a region  $\text{inside}(C)$  of the surface homeomorphic to the plane. If the genus of  $Q$  is greater than 0, call  $\text{inside}(C)$  the *inside region* of  $Q \setminus C$ . For any set  $F \subset V$ , define  $G^F$  to be the *residual graph*, that is, the subgraph of  $G$  induced by those vertices that lie in a dicycle of  $G \setminus F$ .

### 3 Hitting the facial cycles of a digraph

In general, given a (node-weighted) directed graph  $G = (V, E)$  with costs  $c_v \forall v \in V$  a set  $\mathcal{C}$  of cycles of a digraph  $G$ , we define the  $\mathcal{C}$ -hitting set problem as the problem of finding a minimum cost set  $X$  such that  $X \cap C \neq \emptyset \forall C \in \mathcal{C}$ . In this section, we are concerned with when  $\mathcal{C}$  is the set of facial cycles of our graph.

Given a digraph  $G$  embedded on a surface  $Q$ , we show how to obtain an  $O(g)$ -approximation for the problem of finding a minimal hitting set for the set of facial dicycles of a graph embedded on any surface  $Q$  of genus  $g$ .

► **Theorem 2.** *For a graph  $G$  embedded on a surface  $Q$  of genus  $g$ , there is a polynomial-time  $O(g)$ -approximation for the problem of hitting facial dicycles of  $G$ . Moreover, the algorithm returns a DFVS with cost  $O(g)$  times the optimum solution to  $(P_{\text{DFVS}})$  where  $\mathcal{C}$  is the set of facial dicycles.*

*Further, if  $G$  is embedded in a way such that there exist regions  $R_1, R_2$  of  $Q$  homeomorphic to the open disk, such that the inside region of any facial dicycle contains at least one of  $R_1, R_2$ , then there is an algorithm that returns the optimum solution to  $(P_{\text{DFVS}})$  where  $\mathcal{C}$  is the set of facial dicycles.*

**Proof.** We first show that DFVS has an  $O(g)$ -approximation. If  $C$  is a facial cycle bounding a face of  $G$ , call  $C$  *face minimal*. Note that if  $G$  contains a facial cycle, then it must contain a face minimal cycle by the following argument. Let  $C$  be a dicycle such that  $\text{inside}(C)$  is a minimal (by containment) region of  $Q$ . Recall that we removed all vertices of  $G$  not lying on a dicycle. In particular, any vertex  $w$  inside the region  $\text{inside}(C)$  must lie on a facial dicycle  $A_w$ .  $A_w$  cannot be contained entirely in  $\text{cl}(\text{inside}(C))$ , as then the region  $A_w$  would be strictly contained in  $\text{inside}(C)$ . Thus,  $\text{inside}(A_w)$  intersects  $C$  and there is a dipath  $P$  between two nodes  $u, v$  of  $C$ . If  $C$  is not a face, then either there is a vertex  $w$  inside the region  $R_C$  or there is an edge  $uv$  between two nodes  $u, v$  of  $C$  such that  $g(uv) \setminus (g(v) \cup g(u))$  lies in  $R_C$ . In either case, there is a dipath  $P$  between two nodes  $u, v$  of  $C$  then  $P$  together with either the  $u$ - $v$  or  $v$ - $u$  dipath in  $C$  forms a cycle bounding a smaller region of  $Q$ , which is a contradiction.

Our algorithm proceeds as follows. This is a primal-dual algorithm analogous to the technique of [12] for DFVS in planar graphs. Given a feasible dual solution  $y$  to  $(D_{\text{DFVS}})$ , let the *residual cost* of node  $v \in V$  be  $c_v - \sum_{C \in \mathcal{C}, v \in C} y_C$ . For  $\hat{S} \subset V(G)$ , recall  $G^{\hat{S}}$  denotes the subgraph of  $G$  induced by those vertices which are in a dicycle of  $G \setminus \hat{S}$ .

Our primal-dual method begins with a trivial feasible dual solution  $y = \mathbb{0}$ , and the empty, infeasible hitting set  $\hat{S} = \emptyset$ .

While  $G^{\hat{S}}$  contains a facial cycle, increment the dual variables  $y_C$  in  $P_{\text{DFVS}}$  of face minimal cycles  $C$  of  $G$ . When a node of  $G$  becomes tight add it to  $\hat{S}$ . When  $G^{\hat{S}}$  contains no facial cycles apply reverse deletion to  $\hat{S}$  with respect to the facial cycles of  $G^{\hat{S}}$ , that is, we consider

■ **Algorithm 3.1** MinWeightDirectedFVS ( $G, c$ ).

---

**Input** : A digraph  $G = (V, E)$  with non-negative node-costs  $c_v$ , for each  $v \in V$ .  
**Output** : A Directed FVS  $S$  of  $G$ .

```

1  $S = \emptyset$ 
2 while  $G^S$  contains a facial cycle do
3   | Increment all dual variables  $y_C$  for face minimal cycles of  $G^S$ . Add all nodes that
   |   became tight to  $S$ .
4 end while
5 Reverse-Deletion:
6   | Let  $s_1, s_2, \dots, s_l$  be nodes of  $S$  in the order they were added.
7   | for  $t = l$  downto 1 do
8   |   | if  $G^{S \setminus \{s_t\}}$  contains no facial cycle then
9   |   |   |  $S \leftarrow S \setminus \{s_t\}$ 
10  |   | end if
11  |   end for
12
13 return  $S$ 

```

---

each node  $v$  of  $\hat{S}$  in the order it was added and if  $G \setminus (\hat{S} \setminus \{v\})$  contains no facial cycles, delete  $v$  from  $\hat{S}$ . Denote by  $\bar{S}$  the set  $\hat{S}$  at the end of the algorithm. In other words, we apply the primal-dual method to solve the problem of hitting all facial dicycles of  $G$ .

Clearly,  $\bar{S}$  is a feasible hitting set for the set of facial dicycles of  $G$ , we claim it has cost  $O(g)OPT_{LP}$ . To do so we apply that standard analysis of primal-dual methods in [11, 12].

► **Theorem 3** ([11]). *Suppose  $S \subset V(G)$  and  $y$  is a solution to  $(D_{DFVS})$  output by our primal-dual algorithm such that the following holds.*

1.  $y$  is obtained starting with the initial feasible solution  $y := 0$  and incrementing some set of dual variables  $\{y_C : v \in C_t\}$  uniformly and maintaining feasibility of  $y$  for iterations  $t = 1, 2, \dots, l$  for some  $l \in \mathbb{N}$ .
2. For each iteration  $t \in \{1, 2, 3, \dots, l\}$ , the set  $\{y_C : C \in \mathcal{C}_t\}$  of incremented dual variables satisfies  $\sum_{C \in \mathcal{C}_t} |S \cap C| \leq \beta |\mathcal{C}_t|$ .
3.  $\forall v \in S, \sum_{C \in \mathcal{C}} y_C = c_v$ .

Then  $S$  has cost at most  $\beta \sum_{C \in \mathcal{C}} y_C$ , that is at most  $\beta$  times the LP value.

Using Theorem 3, it suffices to prove that during any iteration  $t$ , the face minimal cycles  $\mathcal{C}_t$  of  $G^{S_t}$ , where  $S_t$  is our current hitting set satisfies

$$\sum_{C \in \mathcal{C}_t} |\bar{S} \cap C| \leq O(g) |\mathcal{C}_t|. \quad (2)$$

Again we remove nodes of  $G$  that do not lie on any dicycle. Denote  $\bar{S}_t$  to be the nodes of  $\bar{S}$  that intersect a cycle of  $\mathcal{C}_t$ . So it suffices to show  $\sum_{C \in \mathcal{C}_t} |\bar{S}_t \cap C| \leq O(g) |\mathcal{C}_t|$ .

The following definition of crossing cycles was elementary to the approach by Goemans and Williamson [12].

► **Definition 4.** *Fix an embedding of a planar graph. Two cycles  $C_1, C_2$  cross if  $C_i$  contains an edge intersecting the interior of the region bounded by  $C_{3-i}$ , for  $i = 1, 2$ . That is, the plane curve corresponding to the embedding of the edge in the plane intersects the interior of the region of the plane bounded by  $C_{3-i}$ . A set of cycles  $\mathcal{C}$  is laminar if no two elements of  $\mathcal{C}$  cross.*

Denote  $\mathcal{C}'$  the set of facial cycles of  $\mathcal{C}$ . For a node  $v \in \bar{S}$ , call a cycle  $C \in \mathcal{C}'$  with  $C \cap \bar{S} = \{v\}$  a *witness* for  $v$ . Since we applied reverse deletion to  $\bar{S}$  at the end of the algorithm, each node of  $\bar{S}$  has a witness in  $\mathcal{C}'$  which is a cycle of  $G^S$ .

The following result about the structure of witness cycles was vital to the 3 and 2.4 approximations for DFVS in planar graphs by [12] and [3]. We observe that the proof in [12] which involves iteratively applying an ‘‘uncrossing’’ procedure to two witness cycles that cross yields the same result for facial cycles of graphs on surfaces.

► **Lemma 5** ([12]). *There exists a laminar family  $\mathcal{A} \subset \mathcal{C}'$  of witness cycles in  $G^{S_t}$  for  $\bar{S}_t$ .*

The laminar family  $\mathcal{A}$  can be represented by a forest where  $A_1$  is an ancestor of  $A_2$  if the inside region of  $A_1$  contains the inside region of  $A_2$ . Add a root node  $r$  to this forest, make it the parent of every maximal node of the forest and call the resulting tree  $T$ .

We assign each cycle  $C$  of  $\mathcal{C}_t$  to the smallest node of  $T$  containing  $C$ . Call the set of cycles assigned to  $w \in T$ ,  $\mathcal{C}_w$ . We assign the nodes that  $w$  and the children of  $w$  are witnesses of to  $w$  and call this set  $\bar{S}_w$ .

To bound  $\sum_{C \in \mathcal{C}_t} |\bar{S}_t \cap C|$ , we define the following bipartite graph.

► **Definition 6** ([12]). *The debit graph for  $\mathcal{C}_t$  and  $S$  is the bipartite graph  $\mathcal{D}_G = (\mathcal{R} \cup S, E)$  with edges  $E_{\mathcal{C}_t} = \{(C, s) \in \mathcal{C}_t \times S \mid s \in C\}$ .*

Since each  $C \in \mathcal{C}_t$  is incident to the vertices of  $\bar{S}_t$  on  $C$ ,  $|\bar{S}_t \cap C|$  is the degree of  $C$  in  $\mathcal{D}_G$ . Summing this equality over each  $C \in \mathcal{C}_t$  yields  $\sum_{C \in \mathcal{C}_t} |\bar{S}_t \cap C| = E(\mathcal{D}_G)$ . By placing the node of the debit graph corresponding to  $C$  inside the inside region of  $C$  we can see that the debit graph is also embedded on  $Q$ .

► **Proposition 7** (Corollary of Euler’s formula for graphs of genus  $g$ ). *A (simple) bipartite graph  $\bar{G}$  with at least three vertices embedded on a surface of genus  $g$  satisfies*

$$E(\bar{G}) \leq (2 + g)|V(\bar{G})| - 4$$

if  $\bar{G}$  has two vertices then

$$E(\bar{G}) \leq (2 + g)|V(\bar{G})| - 3$$

**Proof.** Euler’s formula (for instance see [17]) for graphs embedded on a surface of genus  $g$  yields  $2 - 2g = |V(\bar{G})| - |E(\bar{G})| + |F(\bar{G})|$ . Following the same method as the proof of Euler’s formula for bipartite planar graphs with at least 3 vertices, (for instance see Corollary 4.2.10 of [6]) we observe that each face of  $\bar{G}$  having at least 4 edges means  $|F(\bar{G})| \leq \frac{1}{2}|E(\bar{G})|$ . Thus, for  $|V(\bar{G})| \geq 3$ ,  $|E(\bar{G})| \leq 2|V(\bar{G})| - 4 + 4g \leq (2 + g)|V(\bar{G})| - 4$ . If  $|V(\bar{G})| \leq 2$  then  $|E(\bar{G})| \leq 1 \leq (2 + g)|V(\bar{G})| - 3$ . ◀

For a node  $w$  of  $T$  that is not a leaf or the root, the subgraph of  $\mathcal{D}_G$  induced by  $\mathcal{C}_w \cup \bar{S}_w$  is embedded on  $Q$  and further  $|\mathcal{C}_w \cup \bar{S}_w| \geq 3$ , thus by Proposition 7,

$$|E(\mathcal{D}_G(\mathcal{C}_w \cup \bar{S}_w))| \leq (2 + g)|\mathcal{C}_w| + (2 + g)|\bar{S}_w| - 4 = (2 + g)|\mathcal{C}_w| + (2 + g)(\deg_T(w) - 1) - 4. \quad (3)$$

For a leaf  $v$  of  $T$

$$|E(\mathcal{D}_G(\mathcal{C}_v \cup \bar{S}_v))| \leq (2 + g)|\mathcal{C}_v| + 2|\bar{S}_v| - 3 = (2 + g)|\mathcal{C}_v| + 2(\deg_T(v) - 1) - 3. \quad (4)$$

For the root  $r$  of  $T$

$$|E(\mathcal{D}_G(\mathcal{C}_r \cup \bar{S}_r))| \leq (2 + g)|\mathcal{C}_r| + 2|\bar{S}_r| = (2 + g)|\mathcal{C}_r| + 2(\deg_T(r) - 1). \quad (5)$$

Summing these up we get

$$\begin{aligned}
 |E(\mathcal{D}_G)| &= \sum_{v \in T} |E(\mathcal{D}_G(\mathcal{C}_v \cup \bar{S}_v))| \\
 &\leq (2+g)|\mathcal{C}| + \sum_{v \in T} (2+g) \deg_T(v) - 4|T| + l + 4 \\
 &\leq (2+g)|\mathcal{C}| + 2((2+g)|T| - 2) - 4|T| + l + 4 \\
 &\leq (2+g)|\mathcal{C}| + 2g|T| + l \\
 &\leq (3+3g)|\mathcal{C}|
 \end{aligned}$$

where  $l$  is the number of (non-root) leaves of  $T$ . Thus,  $\sum_{C \in \mathcal{C}_t} |\bar{S} \cap C| = |E(\mathcal{D}_G)| \leq (3+3g)|\mathcal{C}|$ .

This shows that  $\bar{S}$  has cost  $O(g)OPT_{LP}$  and hence our algorithm returns a solution of cost  $O(g)OPT_{LP}$ .

Now let us show that in the case  $G$  is embedded in a way such that there exist regions  $R_1, R_2$  of  $Q$  homeomorphic to the open disk, such that the inside region of any facial dicycle contains at least one of  $R_1, R_2$ , then Algorithm 3.1 is an 8-approximation.

The proof works exactly the same as the general case. The key here is to note that the inside regions of face minimal dicycles do not intersect. Thus,  $R_1$  lies in the inside region of at most one cycle in  $\mathcal{C}_t$ . Likewise,  $R_2$  lies in the inside region of at most one cycle in  $\mathcal{C}_t$ . Since inside region of any facial dicycle contains at least one of  $R_1, R_2$ ,  $|\mathcal{C}_t| \leq 2$ . Again Lemma 5 holds. For a facial dicycle  $A$ , denote  $\text{inside}_{\mathcal{C}_t}(A)$  the set of cycles of  $\mathcal{C}_t$  that lie in the closure of the inside region of  $A$ .

► **Lemma 8.** *There do not exist distinct  $A_1, A_2, A_3 \in \mathcal{A}$  such that  $\text{inside}_{\mathcal{C}_t}(A_1) = \text{inside}_{\mathcal{C}_t}(A_2) = \text{inside}_{\mathcal{C}_t}(A_3)$ .*

**Proof.** Suppose such  $A_1, A_2, A_3$  existed. Since they are laminar we may assume w.l.o.g that  $A_1$  is contained in the closure of the inside region of  $A_2$  and  $A_2$  is contained in the closure of the inside region of  $A_3$ . Let  $v_i$  be the hit node that  $A_i$  is the witness of. Note that  $v_2$  does not lie on  $A_1$ . Thus, as  $v_2$  lies outside  $\text{inside}(A_1)$ , it lies outside the closure  $\text{cl}(\text{inside}(A_1))$  of  $\text{inside}(A_1)$ . So  $v_2$  lies in  $Q \setminus \text{inside}(A_3)$ . Thus,  $v_2$  does not lie on any cycle of  $\text{inside}_{\mathcal{C}_t}(A_3)$ . Also  $v_2$  does not lie on  $A_3$ . Thus, as  $v_2$  lies inside  $\text{cl}(\text{inside}(A_3))$ , it lies inside  $\text{inside}(A_3)$ . Thus,  $v_2$  does not lie on any cycle of  $\mathcal{C}_t \setminus \text{inside}_{\mathcal{C}_t}(A_3)$ .

This implies that  $v_2$  does not lie on any cycle of  $\mathcal{C}_t$ , which is a contradiction. ◀

This implies that  $|\bar{S}_t| = |\mathcal{A}| \leq 2(2^{|\mathcal{C}_t|}) \leq 8$ . Thus,  $\sum_{C \in \mathcal{C}_t} |\bar{S}_t \cap C| \leq |\bar{S}_t| |\mathcal{C}_t| \leq 8|\mathcal{C}_t|$ . This shows Algorithm 3.1 is an 8-approximation. ◀

#### 4 Solving the case of no facial cycles

We now show the LP gap of the natural LP ( $P_{DFVS}$ ) for  $G$  has integrality gap  $O(g)$  in the case  $G$  contains no facial cycles. This will allow us to derive an  $O(g)$ -approximation for the general case by first using Theorem 2 to obtain a hitting set  $S$  for the set of facial cycles of cost at most  $O(g)OPT$  and then obtaining a hitting set  $\bar{S}$  for the remaining dicycles.

► **Lemma 9.** *Suppose  $G$  is a digraph embedded on a surface  $Q$  of some fixed genus  $g$  and there is no facial dicycle of  $G$ . Then the LP gap of the natural LP ( $P_{DFVS}$ ) for  $G$  has integrality gap  $O(g)$ .*

**Proof.** We prove the statement by induction on the genus  $g$ . The case  $g = 0$  is trivial because all cycles in planar graphs are facial. Suppose the statement is true for  $g = g'$ . Let  $Q$  be a surface of genus  $g$ , Let  $G$  be a digraph embedded on  $Q$ .

First, while the optimal solution  $\bar{x}$  to  $(P_{\text{DFVS}})$  has a vertex  $v$  with value  $\bar{x}_v \geq \frac{1}{24}$  add  $v$  to our temporary hitting set  $F$ . Formally initialize  $F = \emptyset$ . While the optimal solution  $\bar{x}$  to  $(P_{\text{DFVS}})$  for  $G^F$  contains a value  $\bar{x}_v$  which is  $\frac{1}{24}$  or more add  $v$  to  $F$ .

Let  $F$  denote the final set obtained. Let  $\hat{x}$  be an optimal extreme point solution for the DFVS LP  $(P_{\text{DFVS}})$  for  $G^F$ , so  $\hat{x}_v < \frac{1}{24} \forall v \in V(G^F)$ . Standard results in iterative rounding, see for instance page 14 of [18], show  $F$  has cost at most 24 times the optimal value of our LP.

We now seek to define (integral) distances on  $G^F$ . By standard LP theory,  $\hat{x}$  has rational coordinates. Let  $N \in \mathbb{Z}_{>0}$  be such that  $N\hat{x}$  and  $\frac{1}{12}N$  are integral, call  $N\hat{x}_v$ , the weight of  $v$ . Define the weighted distance of path  $P = v_0, v_1, \dots, v_l$ ,  $\omega(P)$  to be  $\omega(P) := \sum_{i=0}^{l-1} N\hat{x}_{v_i}$ . For a subgraph  $H$  of  $G^F$  define the weighted distance  $d_{\{\omega, H\}}(u, v)$  from  $u$  to  $v$  the minimum weight of a  $u$ - $v$  path in  $H$ . Define  $d_\omega := d_{\{\omega, G\}}$ . For  $U, W \subset V(G)$ , define  $d_{\omega, H}(U, W) := \min_{u \in U, w \in W} d_\omega(u, w)$ . Define  $d_\omega(U, W) = d_{\omega, G}(U, W)$ . Define the weighted distance of a closed walk  $P' = v_0, v_1, \dots, v_l v_0$ ,  $\omega(P')$  to be  $\omega(P') := \sum_{i=0}^l N\hat{x}_{v_i}$ . The results in this paper could also be shown by instead defining the weight of each vertex to be  $\hat{x}_v$  and instead defining the layers (see later) to be the vertices at distance a multiple of  $1/N$  from a given set of vertices. Since  $\hat{x}$  is feasible the following result holds.

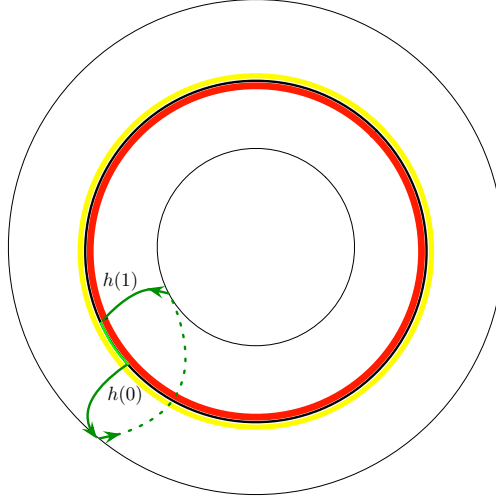
► **Proposition 10.** *The weighted distance of any (directed) closed walk  $P'$  is at least  $N$ .*

Since  $\hat{x}$  is optimal, there exists a dicycle  $C^1 := v_1, v_2, \dots, v_{l'}$  such that  $\sum_{v \in C^1} \hat{x}_v = 1$ . The motivation of our definition of weighted distance comes from [7]. In [7], they also scale the LP values of  $(P_{\text{DFVS}})$  so that the resulting values are integer. For any vertex  $v$  with  $\hat{x}_v = 0$ , they “bypass” the vertex, that is, for each out neighbour  $u$  of  $v$  and in neighbour  $w$  of  $v$ , they add the edge  $wu$  to the graph and when they have done this for all neighbours, they delete  $v$  from the graph. For any vertex  $v$  with  $N\hat{x}_v > 1$  they replace  $v$  by a “chain” of  $N\hat{x}_v > 1$  vertices  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_{N\hat{x}_v}$ , that is, for  $i = 1, 2, \dots, N\hat{x}_v - 1$ ,  $v_i v_{i+1}$  is an edge.  $vv_1$  and  $v_{N\hat{x}_v}u$  are edges for each in neighbour  $w$  and out neighbour  $u$ . Call this graph  $H$ .

For any  $W \subset V(H)$  they define “layers”  $L_i = \{v \in H : d_H(W, v) = i\}$  the nodes at distance  $i$  from  $W$ . They show that the cost of all layers  $L_0, L_1, \dots$  is  $\sum_{v \in V(G)} N\hat{x}_v$ . This is very useful for us as we will use this to show that one layer in  $L_1, \dots, L_m$  has cost at most  $\frac{1}{m} \sum_{v \in V} N\hat{x}_v$ . However, the bypassing operation and replacing a node with a chain operation of [7] do not preserve the genus of the graph. We instead define the notion of weighted distance  $d_\omega$ . Denote the  $i$ -th layer from  $W$  as  $L_i := \{v \in V : i \geq d_\omega(W, v) > i - \omega(v)\}$  the set of nodes for which the distance from  $W$  to  $v$  is at most  $i$ , but for which the distance plus the weight of  $v$  is more than  $i$ . One can see that  $v$  lies in  $N\hat{x}_v$  different  $L_i$ , which is analogous to how  $H$  defined in [7] contains  $N\hat{x}_v$  copies of  $v$  each lying in different layers as well. In particular, a node of weight 0 does not lie in any  $L_i$ , which is analogous to how a vertex of weight 0 is bypassed in [7].

Consider the embedding of  $C^1$  on our surface. Given a subgraph  $W$  of  $G$ , denote by  $g(W)$  the subset of our surface occupied by a vertex or edge of  $W$ . We want to define a “small” neighbourhood around  $g(C^1)$ , not containing any vertices outside  $C^1$ , which we divide up into “left” of  $g(C^1)$  and “right” of  $g(C^1)$ , which we do using the following propositions. These are slightly informal statements of the exact propositions we require, the precise statements appear in Section 5.

► **Proposition 11** (Informal statement of Proposition 23 and Proposition 24). *Given a closed continuous non-self-intersecting curve  $C'$  embedded on an orientable surface  $Q$ , we may partition a small open neighbourhood about  $C'$  into a “left”  $L$  and “right”  $R$ . For any curve  $f : [0, 1] \rightarrow Q$  disjoint from  $C'$  except at  $f(1)$  the partition allows us to say that  $f$  “reaches”  $C'$  from either the left or right.*



■ **Figure 1**  $L$  and  $R$  from Proposition 11 in yellow and red respectively curve  $C'$  depicted in black. The curve  $h$  leaving  $C'$  from the left and entering from the right is depicted in dark green. The closed curve formed by  $h$  and the subcurve of  $C'$  between  $h(0)$  and  $h(1)$  depicted in light green forms a non-facial closed curve.

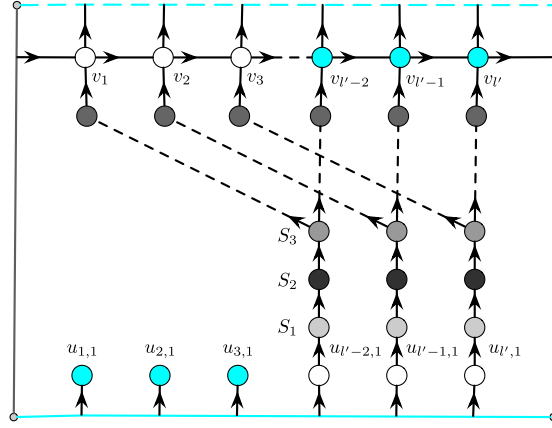
► **Proposition 12** (Informal statement of corollary of Proposition 26). *Let  $C'$  be a non-facial closed curve. If a curve  $h : [0, 1] \rightarrow Q$  “leaves”  $C'$  at a point  $h(0) \in C'$  from the left and reaches  $C'$  at a point  $h(1) \in C'$  from the right, then  $h([0, 1])$  together with a subcurve of  $C'$  from  $h(0)$  to  $h(1)$  is a non-facial closed curve.*

We defer the proofs of Proposition 11 and Proposition 12 for now. We apply Proposition 11 to  $g(C^1)$ . Let  $L, R$  be as in Proposition 11 so that each  $g(e)$  for  $e \in E(G) \setminus E(C^1)$  is disjoint from at least one of  $L, R$  and for each  $e \in E(G \setminus C^1)$ ,  $g(e)$  is disjoint from both  $L, R$ . For each arc  $uv$  of  $G^F$  with exactly one endpoint  $v$  on  $C^1$ ,  $g(uv)$  can be parameterized by a (continuous) curve  $f : [0, 1] \rightarrow g(uv)$  with  $f(0) = g(u)$ ,  $f(1) = g(v)$ . If  $f$  reaches  $g(C^1)$  from the left we say that  $uv$  reaches  $C^1$  from the left, otherwise, we say  $uv$  reaches  $C^1$  from the right.

Let  $u'_{i,1}, u'_{i,2}, \dots, u'_{i,l_i}$  be the out neighbours of  $v_i$  such that the edges  $u'_{i,t'}v_i$  reach  $v_i$  from the left, that is, the arc obtained from reversing the arc  $v_i u'_{i,t'}$  of our graph reaches  $v_i$  from the left. Let  $w'_{i,1}, w'_{i,2}, \dots, w'_{i,z_i}$  be the in neighbours of  $v_i$  such that the edges  $w'_{i,t'}v_i$  reach  $v_i$  from the right. Subdivide each edge  $v_i u'_{i,t}$  into a path  $v_i u_{i,t} u'_{i,t}$  and each edge  $w'_{j,t'} v_j$  into a path  $w'_{j,t'} w_{j,t'} v_j$  and give the new vertices  $w_{j,t'}, u_{i,t}$  infinite cost. There is a natural embedding of our new graph on our surface by placing each  $u_{i,t}$  where the midpoint of the curve  $g(v_i u'_{i,t})$  was embedded and likewise for  $w_{j,t'}$ . By abuse of notation, we continue to call our graph  $G$  and define  $\hat{x}_{u_{i,t}} = \hat{x}_{w_{j,t'}} = 0$  for all  $u_{i,t}, w_{j,t'}$ . Denote  $U := \cup_{i=1}^{l'} \{u_{i,1}, u_{i,2}, \dots, u_{i,l_i}\}$  and  $W := \cup_{i=1}^{l'} \{w_{i,1}, w_{i,2}, \dots, w_{i,z_i}\}$ . For  $X \subset [l']$ , denote  $U_X := \cup_{i \in X} \{u_{i,1}, u_{i,2}, \dots, u_{i,l_i}\}$ ,  $V_X = \{v_i : i \in X\}$  and  $W_X := \cup_{i \in X} \{w_{i,1}, w_{i,2}, \dots, w_{i,z_i}\}$ .

Let  $\tau_- := \{i \in [l'] : \exists w_{i,t'} \in W, \exists u_{j,t} \in U : d_{\omega, G^F \setminus C^1}(u_{j,t}, w_{i,t'}) < \frac{1}{12}N\}$  the first indices of the set of vertices of  $W$  of weighted distance at most  $\frac{1}{12}N$  from  $U$  in  $G^F \setminus C^1$ . Let  $\tau_+ := \{j \in [l'] : \exists u_{j,t} \in U, \exists w_{i,t'} \in W : d_{\omega, G^F \setminus C^1}(u_{j,t}, w_{i,t'}) < \frac{1}{12}N\}$  the first indices of the set of vertices of  $U$  that can reach  $W$  in  $G^F \setminus C^1$  with a path of weighted distance at most  $\frac{1}{12}N$ .

▷ **Claim 13.** If  $d_{\omega}(V_{\tau_-}, V_{\tau_+}) > \frac{1}{12}N$ , then we can find  $S \subset V$ ,  $c(S) = O(1)OPT_{LP}$ , where  $OPT_{LP} := \sum_{v \in V} c_v x_v$  is the value of the optimal fractional solution, such that any strongly connected component of  $G^F \setminus S$  does not contain a directed path from  $U$  to  $W$  in  $G \setminus C^1$ .



■ **Figure 2** Nodes of  $U_{\tau_+}$  and  $V_{\tau_-}$  shown in blue.

If  $d_\omega(V_{\tau_-}, V_{\tau_+}) \leq \frac{1}{12}N$ , then the LP gap of the natural LP (P<sub>DFVS</sub>) for  $G^F$  has integrality gap  $O(g)$ .

Proof. Suppose  $d_{\omega, G^F \setminus C^1}(V_{\tau_-}, V_{\tau_+}) > \frac{1}{12}N$ . For  $i = 0, \dots, \frac{1}{12}N$  let  $S_i := \{v \in V : i \geq d_{\omega, G^F \setminus C^1}(U \setminus U_{\tau_+}, v) > i - \omega(v)\}$  denote the set of vertices of  $V$  that are at weighted distance  $i$  from  $U \setminus U_{\tau_+}$  in  $G^F \setminus C^1$ . (see Figure 2). Since  $d_{\omega, G^F \setminus C^1}(U \setminus U_{\tau_+}, W) > \frac{1}{12}N$ , for  $i = 0, \dots, \frac{1}{12}N$ ,  $W \cap S_i = \emptyset$  and  $W$  is not reachable from  $U \setminus U_{\tau_+}$  in  $(G^F \setminus C^1) \setminus S_i$  for any  $i$ .

Since each  $v$  can lie in at most  $\omega(v)$   $S_i$ ,  $\sum_{i=0}^{\frac{1}{12}N} c(S_i) \leq N \cdot OPT_{LP}$ . Let  $S'$  be the  $S_i$  of minimum cost. For  $i = 0, \dots, \frac{1}{12}N$  let  $T_i := \{v \in V : i > d_{\omega, G^F \setminus C^1}(v, W \setminus W_{\tau_-}) - \omega(v), d_{\omega, G^F \setminus C^1}(v, W \setminus W_{\tau_-}) \geq i\}$ . Since for  $v \in U$ ,  $d_{\omega, G^F \setminus C^1}(v, W \setminus W_{\tau_-}) - \omega(v) = d_\omega(v, W \setminus W_{\tau_-}) > \frac{1}{12}N$ ,  $U \cap T_i = \emptyset$  for  $i = 0, \dots, \frac{1}{12}N$ . Hence  $W \setminus W_{\tau_-}$  is not reachable from  $U$  in  $(G^F \setminus C^1) \setminus T_i$  for any  $i$ . Let  $T'$  be the  $T_i$  of minimum cost.

Finally, let  $Y_i := \{v \in V : i \geq d_\omega(V_{\tau_-}, v) > i - \omega(v)\}$  the set of vertices of weighted distance  $i$  from  $V_{\tau_-}$ . By assumption  $d_\omega(V_{\tau_-}, V_{\tau_+}) > \frac{1}{12}N$  and hence  $V_{\tau_+}$  is not reachable from  $V_{\tau_-}$  in  $G^F \setminus Y_i$  for any  $i = 1, 2, \dots, \frac{1}{12}N$ . Let  $Y'$  be the  $Y_i$  of minimum cost.

Let  $S := S' \cup T' \cup Y'$ . We claim no strongly connected component  $K'$  of  $G^F \setminus S$  contains a directed path from  $U$  to  $W$  in  $G^F \setminus C^1$ . Suppose for a contradiction that some strongly connected component  $K'$  of  $G^F \setminus S$  contains a directed path from some  $u_{i,t} \in U$  to some  $w_{j,t'} \in W$ .

If  $j \notin \tau_-$ , then  $w_{j,t'}$  is not reachable from  $U$  in  $G^F \setminus S$ . If  $i \notin \tau_+$ , then  $W$  is not reachable from  $u_{i,t}$  in  $G^F \setminus S$ . Thus, if either  $j \notin \tau_-$  or  $i \notin \tau_+$  then there is no path from  $u_{i,t}$  to  $w_{j,t'}$  in  $G^F \setminus S$ . Thus,  $j \in \tau_-$  and  $i \in \tau_+$ . As  $K'$  is strongly connected, this implies that  $G^F \setminus S$  contains a path from  $V_{\tau_-}$  to  $V_{\tau_+}$ , which is not possible.

Now suppose that  $d_\omega(V_{\tau_-}, V_{\tau_+}) \leq \frac{1}{12}N$ . Let  $i \in \tau_-$  and  $j \in \tau_+$  be such that  $d_\omega(v_i, v_j) \leq \frac{1}{12}N$ . Let  $P_1, P_2, P_3$  be  $u_{a,t}v_i$ ,  $u_{j,t'}v_j$  and  $v_i v_j$  paths of weight at most  $\frac{1}{12}N$ , with the second last vertices of  $P_1, P_2$  being in  $W$ , for some  $a, b$ . Such paths exist as  $i \in \tau_-$  and  $j \in \tau_+$ . If  $a = i$ , then  $P_1 v_i u_{a,t}$  is a cycle for which  $\sum_{v \in P_1 v_i u_{a,t}} \hat{x}_v < 1$  which is a contradiction. So  $a \neq i$ , likewise  $b \neq j$ .

For  $i', j' \in \{1, 2, \dots, l'\}$ , let  $C_{(v_{i'}, v_{j'})}^1 := v_{i'}, v_{i'+1}, v_{i'+2}, \dots, v_{j'-1} v_{j'}$  (where  $v_t = v_{t \pmod{l'}}$ ) denote the directed path in  $C^1$  from  $v_{i'}$  to  $v_{j'}$ . Note that  $d_\omega(v_{i'}, v_{j'}) = \omega(C_{(v_{i'}, v_{j'})}^1)$ , for otherwise there is a  $v_{i'}v_{j'}$  path  $P'$  of weight less than  $d_\omega(v_{i'}, v_{j'})$ . Then  $C_{(v_{j'}, v_{i'})}^1 \cup P'$  is a directed closed walk of weight  $\omega(C^1) - \omega(C_{(v_{i'}, v_{j'})}^1) + d_\omega(v_{i'}, v_{j'}) < \omega(C^1)$ . Noting that

## 18:10 A Constant Factor Approximation for Directed Feedback Vertex Set

the weighted distance of a cycle is equal to  $N \sum_{v \in C^1} \hat{x}_v$ , we obtain  $N \sum_{v \in C^1_{(v_j, v_{i'})} \cup P'_{i'}} \hat{x}_v < \omega(C^1) = N$ , from which it follows the sum of the  $\hat{x}_v$  values along the closed walk  $C^1_{(v_j, v_{i'})} \cup P'_{i'}$ ,  $\sum_{v \in C^1_{(v_j, v_{i'})} \cup P'_{i'}} \hat{x}_v$  is strictly less than 1, which contradicts the feasibility of  $\hat{x}$ .

We claim  $\omega(C^1_{(v_a, v_i)}), \omega(C^1_{(v_j, v_b)}) \leq \frac{1}{12}N$ . Suppose for a contradiction that  $\omega(C^1_{(v_a, v_i)}) > \frac{1}{12}N$ . Since  $\omega(C^1) = N$ , this implies that  $\omega(C^1_{(v_{i+1}, v_{a-1})}) < N - \frac{1}{12}N$ . Then the cycle  $P_1 C^1_{(v_i, v_a)} v_a u_{a,t}$  satisfies  $\sum_{v \in V(P_1 C^1_{(v_i, v_a)} v_a u_{a,t})} \hat{x}_v < 1 - \frac{1}{12} + \frac{1}{12} = 1$  which is a contradiction. Likewise,  $\omega(v_j, v_b) \leq \frac{1}{12}N$ .

Let us show  $C^1_{(v_a, v_i)} \cap C^1_{(v_j, v_b)} = \{v_i\} \cap \{v_j\}$ , that is the paths  $C^1_{(v_a, v_i)}$  and  $C^1_{(v_j, v_b)}$  are disjoint except in the case  $i = j$  when their intersection is  $v_i$ . First, let us address the case  $i \neq j$ . Suppose for a contradiction that  $C^1_{(v_a, v_i)} \cap C^1_{(v_j, v_b)} \neq \emptyset$ . Let  $v \in C^1_{(v_a, v_i)} \cap C^1_{(v_j, v_b)}$ . Note  $v \neq v_i, v_j$  for otherwise  $C^1_{(v_j, v_i)} P_3$  is a closed walk of weight less than  $N$ . Let  $Q_1$  be a path from  $v$  to  $v_i$  in  $C^1_{(v_a, v_i)}$  and  $Q_2$  a path from  $v_j$  to  $v$  in  $C^1_{(v_j, v_b)}$ . Then  $Q_2 Q_1 P_3$  is a closed walk of weighted distance at most  $\frac{1}{4}N$  which is a contradiction.

Now suppose that  $i = j$ . Suppose for a contradiction that  $C^1_{(v_a, v_i)} \cap C^1_{(v_j, v_b)} \neq \{v_i\}$ . Let  $v \in (C^1_{(v_a, v_i)} \cap C^1_{(v_j, v_b)}) \setminus \{v_i\}$ . Let  $Q_1$  be a path from  $v$  to  $v_i$  in  $C^1_{(v_a, v_i)}$  and  $Q_2$  a path from  $v_i$  to  $v$  in  $C^1_{(v_i, v_b)}$ . Then  $Q_1 Q_2$  is a closed walk of weighted distance at most  $\frac{1}{6}N$ , which is a contradiction.

▷ **Claim 14.**  $d_\omega(P_2 \cup C^1_{(v_{j+1}, v_b)}, P_1 \cup C^1_{(v_a, v_i)}) \geq \frac{1}{6}N$ .

Proof. Suppose for a contradiction that  $d_\omega(P_2 \cup C^1_{(v_{j+1}, v_b)}, P_1 \cup C^1_{(v_a, v_i)}) < \frac{1}{6}N$ . Let  $s \in P_2 \cup C^1_{(v_{j+1}, v_b)}$  and  $q \in P_1 \cup C^1_{(v_a, v_i)}$  be such that  $d_\omega(s, q) < \frac{1}{6}N$ . Let  $P'_1$  be the directed path in  $P_1 \cup C^1_{(v_a, v_i)}$  from  $q$  to  $v_i$ .  $P'_2$  the directed path in  $P_2 \cup C^1_{(v_j, v_b)}$  from  $v_j$  to  $s$  and  $Q$  a path of weight at most  $\frac{1}{6}N$  from  $s$  to  $q$ . Then  $\bar{C} := v_j P'_2 Q P'_1 P_3$  is a closed walk such that  $\sum_{v \in \bar{C}} \hat{x}_v < 1$  which is a contradiction (see Figure 3).

Thus,  $d_\omega(P_2 \cup C^1_{(v_i, v_b)}, P_1 \cup C^1_{(v_a, v_i)}) \geq \frac{1}{6}N$ . Since  $d_\omega(u, v_i) \leq \frac{1}{12}N$  for any  $u \in C^1_{(v_a, v_i)}$ ,  $d_\omega(P_2, C^1_{(v_a, v_i)}) \geq \frac{1}{12}N$   $\triangleleft$

For  $i = 0, 1, \dots, \frac{1}{12}N$ , define  $R_i := \{v \in V : i \geq d_\omega(P_2 \cup C^1_{(v_{j+1}, v_b)}, v) > i - \omega(v)\}$ . Each vertex  $v \in V$  lies in at most  $\omega(v)$   $R_i$ . Let  $R'$  be the  $R_i$  of the smallest cost, so  $c(R') \leq 12OPT_{LP}$ . Since  $d_\omega(P_2 \cup C^1_{(v_{j+1}, v_b)}, P_1 \cup C^1_{(v_a, v_i)}) \geq \frac{1}{12}N$ , it follows that  $(P_1 \cup C^1_{(v_a, v_i)}) \setminus R_i$  is not reachable from  $(P_2 \cup C^1_{(v_{j+1}, v_b)}) \setminus R_i$  in  $G \setminus R_i$  for any  $i$ . Thus,  $(P_1 \cup C^1_{(v_a, v_i)}) \setminus R'$  is not reachable from  $(P_2 \cup C^1_{(v_{j+1}, v_b)}) \setminus R'$  in  $G \setminus R'$ .

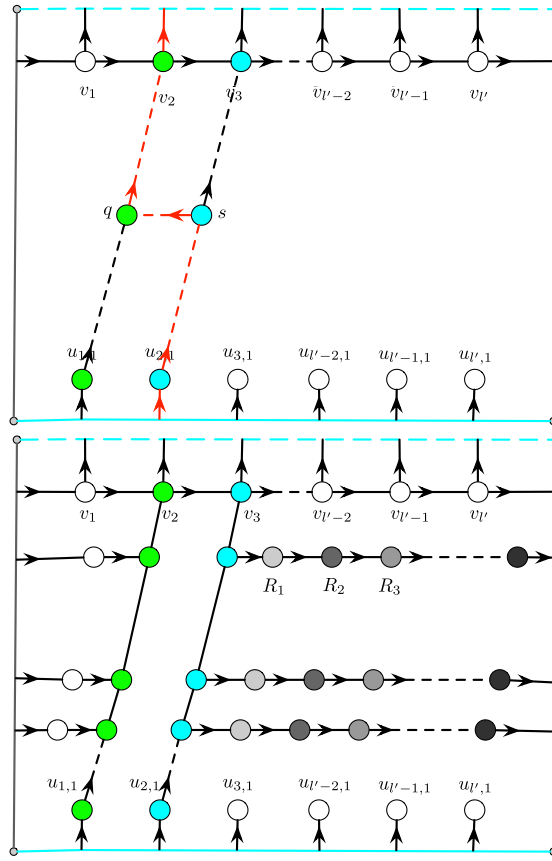
Thus, any strongly connected component of  $G \setminus R'$  is either contained in  $G^F \setminus (P_1 \cup C^1_{(v_a, v_i)})$  or  $G^F \setminus (P_2 \cup C^1_{(v_{j+1}, v_b)})$ . For  $i = 1, 2, \dots, \frac{1}{12}N$  let  $K_i^+ := \{v \in V : i \geq d_\omega(v_j, v) > i - \omega(v)\}$  be the vertices of weighted distance  $i$  from  $v_j$ . Let  $K'^+$  denote the  $K_i^+$  of minimum cost.

▷ **Claim 15.**  $v_j$  is not contained in a cycle in  $G^F \setminus (R' \cup K'^+)$ .

Proof. Suppose that there is a cycle  $a_1, a_2, \dots, a_p v_j a_1$  in  $G^F \setminus (R' \cup K'^+)$ . If  $d_\omega(v_j, a_p) > \frac{1}{12}N$ , then  $a_p$  is not reachable from  $v_j$  in  $G^F \setminus (R' \cup K'^+)$ . Thus, there is a path  $P_a$  of weighted distance at most  $\frac{1}{12}N$  from  $v_j$  to  $a_p$ . Then the closed walk  $v_j P_a a_p v_j$  has weighted distance at most  $\omega(P_a) + \omega(a_p) \leq \frac{1}{12}N + \frac{1}{12}N < N$  which is a contradiction.  $\triangleleft$

Recall that any dicycle of  $G^F \setminus R'$  is contained in either  $G^F \setminus (P_1 \cup C^1_{(v_a, v_i)})$  or  $G^F \setminus (P_2 \cup C^1_{(v_{j+1}, v_b)})$ . Since  $v_j$  is not contained in any dicycle of  $G^F \setminus (R' \cup K'^+)$  it follows that any dicycle of  $G^F \setminus (R' \cup K'^+)$  is either contained in  $G \setminus (P_1 \cup C^1_{(v_a, v_i)})$  or in  $G \setminus (P_2 \cup C^1_{(v_j, v_b)})$ . By Proposition 12,  $g(P_1 \cup C^1_{(v_a, v_i)})$  and  $g(P_2 \cup C^1_{(v_j, v_b)})$  are nonfacial.





■ **Figure 3** On the left, there are  $u_{1,1}-v_2$  and  $u_{2,1}-v_3$  paths (green and blue vertices respectively) of weight at most  $\frac{1}{12}N$  and  $s-q$  path of length at most  $\frac{1}{12}N$ . The red cycle would then have weight at most  $N$ , which is a contradiction. On the right are the sets  $R_i$ , vertices at distance  $i$  from  $P_2 \cup C^1_{(v_{j+1}, v_b)}$ .

► **Definition 16** ([1, 14]). Given a simple closed curve  $f$  on a surface without boundary  $Q$ , not dividing the surface into 2 regions, we say  $Q'$  is obtained by doing surgery along  $f$  if  $Q'$  is obtained as follows. “Thicken”  $f$  to obtain a cylinder and remove this cylinder from  $Q$ , call this resulting surface  $Q''$ . The boundary of  $Q''$  consists of 2 circles we “glue” two cones  $N_1, N_2$  along these circles and call this final surface  $Q'$ .

► **Theorem 17** ([1] p.162). For a surface without boundary  $Q$  of genus  $g'$ ,  $Q'$  obtained by Definition 16 is a surface without boundary of genus at most  $g' - 1$ .

We apply the surgery of Definition 16 to  $g(P_1 \cup C^1_{(v_a, v_i)})$  to obtain a surface  $Q'$  of genus one less than  $Q$ . Let  $N'_1, N'_2$  denote the two cones glued to  $Q'$ . We also apply the surgery of Definition 16 to  $g(P_2 \cup C^1_{(v_j, v_b)})$  to obtain a surface  $\hat{Q}$  of genus one less than  $Q$ . Let  $\hat{N}_1, \hat{N}_2$  denote the two cones glued to  $\hat{Q}$ .

► **Lemma 18.** Let  $G$  be a graph embedded on a surface  $Q$  with no dicycles. Let  $h$  be a non-facial curve of  $Q \setminus G$ . Let  $Q'$  be the surface obtained by applying the surgery of Definition 16 to with respect to the curve  $h$  and surface  $Q$ . There is a natural embedding of  $G$  on  $Q'$  (by leaving each node of  $G$  where it was in  $Q$ ). Let  $N_1, N_2$  denote the two cones glued to  $Q'$  during the surgery process. Then each facial cycle of  $G$  with respect to its embedding in  $Q'$  contains either  $N_1$  or  $N_2$  in its inside region.

## 18:12 A Constant Factor Approximation for Directed Feedback Vertex Set

**Proof.** Let  $C$  be a facial cycle of  $G^F \setminus (P_1 \cup C_{(v_a, v_i)}^1)$  with respect to its embedding in  $Q'$ . If neither of the cones  $N_1, N_2$  are contained in the inside region of  $C$ , then  $C$  is a facial cycle of  $G$  with respect to its embedding in  $Q$ , which is a contradiction.  $\blacktriangleleft$

Thus, any facial cycle of  $G^F \setminus (P_1 \cup C_{(v_a, v_i)}^1)$  contains either  $N'_1$  or  $N'_2$  in its inside region.

Now let  $G_1, G_2, \dots, G_l$  be the strongly connected components of  $G^F \setminus (R' \cup K'^+)$ . Since any closed walk of  $G^F \setminus (R' \cup K'^+)$  is either contained in  $G \setminus (P_1 \cup C_{(v_a, v_i)}^1)$  or in  $G \setminus (P_2 \cup C_{(v_j, v_b)}^1)$  each strongly connected component is either contained in  $G \setminus (P_1 \cup C_{(v_a, v_i)}^1)$  or in  $G \setminus (P_2 \cup C_{(v_j, v_b)}^1)$ . If  $G_i$  is contained in  $G \setminus (P_1 \cup C_{(v_a, v_i)}^1)$ , then there is a natural embedding of  $G_i$  in  $Q'$  (obtained by leaving all nodes and edges where they are in the surgery for Definition 16). Likewise, if  $G_i$  is contained in  $G \setminus (P_2 \cup C_{(v_j, v_b)}^1)$ , then there is a natural embedding of  $G_i$  in  $\hat{Q}$ . Thus, for any  $G_i$  contained in  $G \setminus (P_1 \cup C_{(v_a, v_i)}^1)$  by Theorem 2, there is an 8-approximation for the problem of hitting the facial cycles of  $G_i$  (with respect to the natural embedding in  $Q'$ ). Likewise, for any  $G_i$  contained in  $G \setminus (P_2 \cup C_{(v_j, v_b)}^1)$  there is an 8-approximation for the problem of hitting the facial cycles of  $G_i$ . Let  $Z_i$  be a solution for the problem of hitting facial cycles of  $G_i$  of cost at most  $8OPT_{LP(G_i)}$  as guaranteed by Theorem 2.

Then each  $G_i \setminus Z_i$  is embedded in a surface of smaller genus with no facial cycles.

By induction, there are solutions  $A_i$  to  $G_i \setminus Z_i$  of cost  $c_{g-1}OPT_{LP(G_i \setminus Z_i)}$ , where  $c_g$  is the integrality gap of the DFVS LP for graphs of genus  $g$ .

Define  $\hat{x}^{G_i \setminus Z_i} \in \mathbb{R}^{V(G_i \setminus Z_i)}$  as  $\hat{x}^{G_i \setminus Z_i}(v) = \hat{x}_v$ , where  $\hat{x}$  is as in the proof of Lemma 9. Since graphs  $G_i$  are vertex disjoint,  $G_i \setminus Z_i$  are vertex disjoint, so  $\sum_{i=1}^l OPT_{LP(G_i)} \leq \sum_{i=1}^l \sum_{v \in V(G_i)} \hat{x}_v \leq \sum_{v \in V(G)} \hat{x}_v = OPT_{LP(G)}$ . Now  $F \cup R' \cup K'^+ \cup (\cup_{i=1}^l A_i) \cup (\cup_{i=1}^l Z_i)$  is a DFVS of cost  $(O(1) + c_{g-1})OPT_{LP(G)} = (O(1) + O(g-1))OPT_{LP(G)} = (O(g))OPT_{LP(G)}$ .  $\triangleleft$

Note that the argument in Claim 13 is symmetric with respect to left and right and we may swap right and left to get the following result. Let  $b'_{i,1}, b'_{i,2}, \dots, b'_{i,l'_i}$  be the in neighbours of  $v_i$  such that each edge  $b'_{i,t}v_i$  reaches  $v_i$  from the left and  $d'_{i,1}, d'_{i,2}, \dots, d'_{i,t'_i}$  be the out neighbours of  $v_i$  such that the edge  $d'_{i,t'}v_i$  reaches  $v_i$  from the right. Subdivide each edge  $b'_{j,t'}v_j$  into a path  $b'_{j,t'}b_{j,t'}v_j$  and each edge  $v_i d'_{i,t}$  into a path  $v_i d_{i,t} d'_{i,t}$  and give the new vertices  $d_{j,t'}, b_{i,t}$  infinite cost. There is a natural embedding of our new graph on our surface by placing each  $b_{i,t}$  where the midpoint of the curve  $g(v_i b'_{j,t'})$  was embedded and likewise for  $d_{j,t'}$ . By abuse of notation, we continue to call our graph  $G$  and define  $\hat{x}_{b_{i,t}} = \hat{x}_{d_{j,t'}} = 0$  for all  $b_{i,t}, d_{j,t'}$ .

Denote  $B := \cup_{i=1}^{l'} \{b_{i,1}, b_{i,2}, \dots, b_{i,l'_i}\}$ ,  $D = \cup_{i=1}^{l'} \{d_{i,1}, d_{i,2}, \dots, d_{i,t'_i}\}$ . Let  $\kappa_- : \{i \in [l'] : \exists b_{i,t'} \in B, \exists d_{j,t'} \in D : d_{\omega, G^F \setminus C^1}(d_{j,t'}, b_{i,t'}) < \frac{1}{12}N\}$  the first indices of the set of vertices of  $B$  of weighted distance at most  $\frac{1}{12}N$  from  $D$  in  $G^F \setminus C^1$ . Let  $\kappa_+ := \{j \in [l'] : \exists d_{j,t'} \in D, \exists b_{i,t'} \in B : d_{\omega, G^F \setminus C^1}(d_{j,t'}, b_{i,t'}) < \frac{1}{12}N\}$  the first indices of the set of vertices of  $D$  that can reach  $B$  with a path of weighted distance at most  $\frac{1}{12}N$  in  $G^F \setminus C^1$ . Similarly to how we proved Claim 13, we can show the following:

$\triangleright$  **Claim 19.** If  $d_{\omega}(V_{\kappa_-}, V_{\kappa_+}) > \frac{1}{12}N$ , then we can find  $T \subset V$ ,  $c(T) = O(1)OPT_{LP}$ , (recall  $OPT_{LP} := \sum_{v \in V} c_v x_v$  is the value of the optimal fractional solution), such that any strongly connected component of  $G^F \setminus T$  does not contain a directed path from  $D$  to  $B$  in  $G \setminus C^1$ .

If  $d_{\omega}(V_{\kappa_-}, V_{\kappa_+}) \leq \frac{1}{12}N$ , then the LP gap of the natural LP ( $P_{DFVS}$ ) for  $G$  has integrality gap  $O(1)$ .

We now construct a DFVS of cost at most  $O(g)OPT_{LP}$ . If either  $d_{\omega}(V_{\kappa_-}, V_{\kappa_+}) \leq \frac{1}{12}N$  or  $d_{\omega}(V_{\tau_-}, V_{\tau_+}) \leq \frac{1}{12}N$ . Then Claim 13 or Claim 19 respectively shows that the LP gap of the natural LP ( $P_{DFVS}$ ) for  $G$  has integrality gap  $O(g)$ .

Now assume both  $d_\omega(V_{\tau_-}, V_{\tau_+}), d_\omega(V_{\kappa_-}, V_{\kappa_+}) > \frac{1}{12}N$ . Then by Claim 13 and Claim 19, there are sets  $S, T$  such that any strongly connected component of  $G^F \setminus (S \cup T)$  does not contain a path from  $U$  to  $W$  or a path from  $D$  to  $B$  in  $G^F \setminus C^1$ .

For any digraph  $H$  define  $\text{un}(H)$  to be the underlying (undirected) graph of  $H$ . Let  $K$  be any strongly connected component of  $G^F \setminus (S \cup T)$ . We will prove  $\text{un}(K)$  does not contain any path from  $U \cup B$  to  $W \cup D$  in  $\text{un}(K) \setminus C^1$ .

► **Proposition 20.** *If there is a (undirected) path  $P = u_{i,t}q_1, q_2, \dots, q_t$  from some  $u_{i,t} \in U$  (resp  $u_{i,t} \in D$ ) in  $\text{un}(K) \setminus C^1$ , then there is a directed path from  $U$  (resp  $D$ ) to  $q_j$  in  $G^F \setminus (S \cup T \cup C^1)$  for any  $j = 1, 2, \dots, t$ .*

*If there is a (undirected) path  $P = q_1, q_2, \dots, q_t w_{i,t}$  from some  $w_{i,t} \in W$  (resp  $b_{i,t} \in B$ ) in  $\text{un}(K) \setminus C^1$ , then there is a directed path from  $q_j$  to  $W$  (resp  $B$ ) in  $G^F \setminus (S \cup T \cup C^1)$  for any  $j = 1, 2, \dots, t$ .*

**Proof.** Let  $P = u_{i,r}q_1, q_2, \dots, q_t$  be a path in  $\text{un}(K) \setminus C^1$  from some  $u_{i,r} \in U$  (resp  $u_{i,r} \in D$ ). We prove by induction  $t'$  on that there is a directed path from  $U$  to  $q_j$  in  $G^F \setminus (S \cup T \cup C^1)$  for any  $j = 1, 2, \dots, t'$ . The case  $t' = 1$  is clear as each  $u_{i,r} \in U$  (resp  $u_{i,r} \in D$ ) only has out-neighbours so the undirected edge  $\{u_{i,r}, q_1\}$  in  $\text{un}(K)$  is directed from  $u_{i,r}$  to  $q_1$ .

Now assume the statement true for  $t' = t''$ . For  $t' = t'' + 1$ , if the undirected edge  $\{q_{t'}, q_{t'+1}\}$  is directed from  $q_{t'}$  to  $q_{t'+1}$ , then there is a directed path from  $u_{i,r}$  to  $q_{t'+1}$  in  $G^F \setminus (S \cup T \cup C^1)$ .

Otherwise  $\{q_{t'}, q_{t'+1}\}$  is directed from  $q_{t'+1}$  to  $q_{t'}$ . By strong connectedness of  $K$ , there is a directed path  $P'$  from  $q_{t'}$  to  $q_{t'+1}$  in  $K \setminus (S \cup T)$ . If  $P'$  does not intersect  $C$  then there is a directed path from  $u_{i,r}$  to  $q_{t'+1}$  in  $G^F \setminus (S \cup T \cup C^1)$ . So, assume  $P'$  intersects  $W$  or  $B$ . Let  $P''$  denote the subpath of  $P'$  from  $q_{t'}$  to when  $P'$  first intersects  $U$  or  $B$ . By construction  $P''$  lies in  $G^F \setminus (S \cup T \cup C^1)$ . As  $u_{i,r}$  lies in  $U$  (resp  $D$ )  $P''$  does not intersect  $W$  (resp.  $B$ ), as then we would have a  $U$ - $W$  (resp.  $D$ - $B$ ) path in  $G^F \setminus (S \cup T \cup C^1)$ . Thus,  $P''$  is a  $q_{t'}$ - $B$  (resp.  $q_{t'}$ - $W$ ) path. Consider the subpath  $Q$  of the reversal of  $P'$  starting from  $q_{t'+1}$  to when the reversal of  $P'$  first intersects  $D$  or  $U$ . Let  $\text{rev}(Q)$  denote the reversal of  $Q$ . Note  $\text{rev}(Q)$  lies in  $G^F \setminus (S \cup T \cup C^1)$ . If the starting vertex of  $\text{rev}(Q)$  is in  $D$  (resp.  $U$ ), then  $\text{rev}(Q) \cup \{q_{t'+1}q_{t'}\} \cup P''$  is a  $D$ - $B$  (resp.  $U$ - $W$ ) path in  $G^F \setminus (S \cup T \cup C^1)$ . This contradicts Claim 19. Thus, the starting vertex of  $\text{rev}(Q)$  is in  $U$  (resp.  $D$ ). This implies there is a path from  $U$  (resp.  $D$ ) to  $q_{t'+1}$  completing the induction. The proof of the second part is similar. ◀

► **Proposition 21.** *There is no (undirected) path from  $W \cup D$  to  $U \cup B$  in  $\text{un}(K) \setminus C^1$ .*

**Proof.** If we have a  $U$ - $W$  path  $P = u_{i,t}q_1, q_2, \dots, q_t w_{j,t'}$  in  $\text{un}(K) \setminus C^1$ , then by Proposition 20, there are directed  $U$ - $q_1$  and  $q_1$ - $W$  paths  $P_1$  and  $P_2$  in  $\text{un}(K) \setminus C^1$ . Then  $P_1 \cup P_2$  is a directed  $U$ - $W$  path in  $K \setminus C^1$  which contradicts Claim 19. Thus, we do not have a  $U$ - $W$  path  $P = u_{i,t}q_1, q_2, \dots, q_t w_{j,t'}$  in  $K \setminus C^1$ . Likewise, we do not have a  $D$ - $B$  path  $P = u_{i,t}q_1, q_2, \dots, q_t w_{j,t'}$  in  $K \setminus C^1$ .

Suppose we have a  $U$ - $D$  path  $P = u_{i,t}q_1, q_2, \dots, q_t d_{j,t'}$  in  $\text{un}(K) \setminus C^1$ . By Proposition 20, there are directed  $U$ - $q_1$  and  $D$ - $q_1$  paths  $P_1$  and  $P_2$  in  $K \setminus C^1$ . Recall  $U$  has no in-neighbours of in  $G \setminus C^1$ , so the edge  $\{u_{i,t}, q_1\}$  in  $K$  is directed from  $u_{i,t}$  to  $q_1$ . By 2 connectedness of  $K$ , there is a path  $P_3$  from  $q_1$  to  $u_{i,t}$ . The only in-neighbours of  $u_{i,t}$  are in  $C^1$ , thus  $P_3$  intersects  $W \cup B$ . Let  $P'_3$  be the subpath of  $P_3$  from  $q_1$  to when it the path first intersects  $W \cup B$ . If the endpoint of  $P'_3$  is in  $W$ , then  $P_1 \cup P'_3$  is a  $U$ - $W$  path in  $K \setminus C^1$ . Otherwise, if the endpoint of  $P'_3$  is in  $B$ , then  $P_2 \cup P'_3$  is a  $D$ - $B$  path in  $K \setminus C^1$ . Either way this contradicts Claim 19. ◀

► **Proposition 22** ([15, 25]). *Suppose  $G$  is a graph embedded on a surface  $Q$ . Let  $C$  be a cycle of  $G$  that does not divide  $Q$  into two separate regions such that there is no edge between vertices of  $C$  that is not part of  $C$ . Define a “left” and “right” as in Proposition 11. Let  $\hat{L}, \hat{R}$  denote the neighbours of  $C$  that are “left” or “right” of  $C$ . Suppose each connected component of  $G \setminus C$  only contains nodes of  $\hat{L}$  or  $\hat{R}$  but not both. There is a non-facial closed curve  $h$  in  $Q \setminus G$ .*

Applying Proposition 21, we get that  $G^F$  satisfies Proposition 22 with respect to  $C^1$ . Thus, there is a non-facial closed curve  $h$  in  $Q \setminus G^F$ . We apply the surgery of Definition 16 with respect to the closed curve  $h$  and surface  $Q$  to obtain a surface  $Q'$  of lower genus. Let  $G_1, G_2, \dots, G_l$  be the strongly connected components of  $G^F \setminus (S \cup T)$ , so each  $G_i$  is embeddable on  $Q'$ . By Lemma 18 each facial dicycle of  $G_i$  contains one of the cones of  $Q'$ . Hence there is an algorithm that returns a hitting set  $Z_i$  to the set of facial cycles of  $G_i$  of cost at most  $8OPT_{LP(G_i)}$ . By induction, there are solutions  $A_i$  to  $G_i \setminus Z_i$  of cost  $c_{g-1}OPT_{LP(G_i)}$ , where  $c_g$  is the integrality gap of the DFVS LP for graphs of genus  $g$ . Define  $\hat{x}_i^G \in \mathbb{R}^{V(G_i)}$  as  $\hat{x}_i^G(v) = \hat{x}_v$ , where  $\hat{x}$  is as in the proof of Lemma 9. Since graphs  $G_i$  are vertex disjoint,  $\sum_{i=1}^l OPT_{LP(G_i)} \leq \sum_{i=1}^l \sum_{v \in V(G_i)} \hat{x}_i^G(v) \leq \sum_{v \in V(G)} \hat{x} = OPT_{LP(G)}$ . Then  $S \cup T \cup F \cup (\cup_{i=1}^l A_i) \cup (\cup_{i=1}^l Z_i)$  is a DFVS of cost  $(O(1) + c_{g-1})OPT_{LP(G)} = (O(1) + O(g-1))OPT_{LP(G)} = (O(g))OPT_{LP(G)}$ . ◀

As observed in [7],  $(P_{DFVS})$  can be solved in polynomial-time via the ellipsoid method. Hence Lemma 9 yields a polynomial time  $O(g)$ -approximation algorithm for DFVS in graphs of genus  $g$  with no facial cycle.

## 5 Statement and proofs of topological results we use

First let us prove Proposition 22.

**Proof.** Suppose each connected component of  $G \setminus C$  only contains nodes of  $\hat{L}$  or  $\hat{R}$  but not both. Let  $G_L$  and  $G_R$  be the unions of the components of  $G - C$  that only contain nodes from  $\hat{L}$  and  $\hat{R}$  respectively. Assume that  $Q \setminus G$  contains no non-facial curve  $h$ .

Case 1: At least one of  $G_L$  or  $G_R$  is empty.

Suppose, without loss of generality, that  $G_L$  is empty. Consider the face of  $G$  that contains  $C$  and intersects the left of  $C$ . But,  $C$  is not contractible (else it would separate the surface  $Q$  into two components). Hence, a small leftward shift of  $C$  which will lie in the face  $f$  will produce a non-facial curve  $h$ .

Case 2: Both  $G_L$  and  $G_R$  are nonempty.

We claim that if a face contains vertices of  $G_L, G_R$  and of  $C$  then there is a non-facial curve in  $Q \setminus G$ . Let  $f$  be such a face of degree  $d$ . Let  $\partial f = v_0 \cdots v_{d-1}$  be the boundary cycle of  $f$ , where  $i \in \mathbb{Z}/d\mathbb{Z}$ . Without loss of generality, assume that  $v_0 \in \hat{L}$  and for some  $q$   $v_1, v_2, \dots, v_q \in C$ , and  $v_{q+1} \in \hat{R}$ . There are points  $p_L$  on the edge  $v_0 v_1$  in the interior of  $L$  and  $p_R$  on the edge  $v_q v_{q+r}$  in the interior of  $R$ . Let  $h : [0, 1] \rightarrow Q$  be a non-self-intersecting curve in  $f$  from  $p_R$  to  $p_L$ . Let  $r_L, r_R > 0$  be such that  $B_Q(p_L, r_L) \subset L$ ,  $B_Q(p_R, r_R) \subset R$ . Let  $h^L, h^R$  be non-self-intersecting curves in  $B_Q(p_L, r_L)$  and  $B_Q(p_R, r_R)$  from  $p_L$  to  $v_1$  and  $p_R$  to  $v_q$  respectively not intersecting  $h$ . Then  $h \cup h^L \cup h^R$  satisfies the conditions of Proposition 12. Thus,  $h \cup h^L \cup h^R \cup g(p_L v_0, v_1, \dots, v_{q+1} p_R)$  does not bound a region of the closure of  $f$ . As this curve lies in the closure of  $f$ , this implies that  $f$  is not homeomorphic to an open disk. By the classification theorem for orientable surfaces (see for instance page 87 of [17]),  $cl(f)$  is homeomorphic to a  $m$ -torus  $T_m$  with a finite number of open disks removed. Since  $f$  is not homeomorphic to an open disk,  $f$  contains a non-facial closed curve  $h$  in its interior.

If there is a face  $f$  of  $G$  whose boundary contains vertices of  $G_L$  and of  $G_R$  (but not of  $C$ ), then as there is no edge between  $G_L$  and  $G_R$ , the boundary of  $f$  is not connected and so  $f$  is not homeomorphic to an open disk. Just as before this implies  $f$  contains a non-facial closed curve  $h$ .

Now, consider the subsets  $Q_L$  and  $Q_R$  of  $Q$  obtained by taking the union of all the vertices, edges and faces induced by  $G_L \cup C$  and  $G_R \cup C$ , respectively. By assumption 3, every component of  $G - C$  is in  $G_L$  or  $G_R$ , so every vertex and edge of  $G$  belongs to  $Q_L$  or  $Q_R$ . By the subcases eliminated above under Case 2, every face of  $G$  also belongs to  $Q_L$  or  $Q_R$  (but not both). Then,  $Q = Q_L \cup Q_R = (Q_L - C) \sqcup (Q_R - C) \sqcup C$ . This means that  $C$  separates  $Q$  into two components, which contradicts assumption 1. ◀

It is well known (see for instance [1] page 15) that smooth surfaces  $Q$  have the property that for each  $v \in Q$  there is an open ball  $B_Q(v, r_0)$  of some small radius  $r_0 > 0$  in  $Q$  and a diffeomorphism  $\psi$  from  $B_Q(v, r_0)$  to the open disk  $B_{\mathbb{R}^2}(0, r_0)$  of radius  $r_0$  about the origin in the two-dimensional plane such that  $\psi(v) = (0, 0)$  and  $\psi$  preserves distances from  $v$ , that is  $\text{dist}_Q(v, x) = \|\psi(v) - \psi(x)\|$ , where  $\text{dist}_Q(v, x)$  is the geodesic distance from  $v$  to  $x$  in  $Q$ . For  $p \in Q$ ,  $r > 0$ , denote by  $B(p, r)$  the open ball of radius  $r$  about  $p$ . We now formally state and prove what Proposition 11 and Proposition 12 informally say.

► **Proposition 23.** *Given a closed continuous non-self-intersecting curve  $C'$  embedded on an orientable surface  $Q$ . There exist some radius  $r > 0$  and disjoint subsets  $L, R$  “on each side” of  $C'$  such that the set  $\{B(v, r) : v \in C'\}$  (where  $B(v, r)$  is the open ball around  $v$  of radius  $r$  in  $Q$ ) is contained in the union  $L \cup R \cup C'$ , and for each  $v \in C'$ ,  $r' \leq r$ ,  $L \cap B(v, r')$  and  $R \cap B(v, r')$  are the two connected components of  $B(v, r') \setminus C'$ . There is a diffeomorphism  $\phi$  from  $L \cup C' \cup R$  to a connected open neighbourhood of  $C' \times \{0\}$  in  $C' \times \mathbb{R}$  and small  $q > 0$  with  $C' \times (-q, 0) \subset \phi(L) \subset C' \times (-\infty, 0)$ ,  $C' \times (0, q) \subset \phi(R) \subset C' \times (0, \infty)$  and  $\phi(C') = C' \times \{0\}$ . Further for any (piecewise smooth) curve  $f : [0, 1] \rightarrow Q$  such that  $f(x) \notin C'$  for any  $x \in [0, 1)$ ,  $f(1) \in C'$  satisfies that for some  $\beta \in (0, 1)$ , either  $f((\beta, 1)) \in L$ , that is the curve “reaches  $C'$  from the left”  $L$  or  $f((\beta, 1)) \in R$ , that is the curve “reaches  $C'$  from the right”  $R$ .*

► **Proposition 24.** *For a finite set of curves  $f_1, f_2, f_3, \dots, f_{t'}, h_1, h_2, \dots, h_{t'} : [0, 1] \rightarrow Q$  such that for each  $i$ ,  $f_i(x) \notin C'$  for any  $x \in [0, 1)$  and  $h_i(x) \notin C'$  for any  $x \in [0, 1]$  we may choose  $L, R, r$  above so that each curve  $f_i([0, 1))$  is disjoint from at least one of  $L, R$  and each curve  $h_i([0, 1])$  is disjoint from both  $L, R$ . We refer to  $L$  and  $R$  as the left and right of  $C'$  respectively.*

Further, there are curves  $f_L : [0, 1] \rightarrow L$ ,  $f_R : [0, 1] \rightarrow R$  which are homotopic to  $C'$ . Informally speaking, these are obtained by “slightly shifting”  $f$  “left” and “right” respectively.

► **Proposition 25.** *Lastly let  $h : [0, 1] \rightarrow Q$  be any curve that reaches  $C'$  from the right at a point  $c_2 = h(1)$  on  $C'$ , leaves  $C'$  from the left at  $c_1 = h(0)$ , that is the curve  $\bar{\psi}(t) = h(1 - t)$  reaches  $C'$  at  $c_1$  from the left and  $h$  is otherwise disjoint from  $C'$ . Assume  $c_1 \neq c_2$  and let  $C'_{c_1, c_2}$  be a subcurve of  $C'$  with endpoints  $c_1$  and  $c_2$ . Then there is a curve  $\hat{h} : [0, 2] \rightarrow Q$  that reaches  $C'$  from the right at  $c_1 = \hat{h}(1)$  and leaves  $C'$  at a point  $c_1 = \hat{h}(0)$   $\hat{h}$  is otherwise disjoint from  $C'$ , and there is a homeomorphism of  $Q$  that maps  $\hat{h}$  to the concatenation of  $h$  and  $C'_{c_1, c_2}$ .*

► **Proposition 26.** *Let  $Q$  be an orientable surface and  $\phi : [0, 1] \rightarrow Q$  a closed curve not dividing  $Q$  into 2 regions with disjoint subsets  $L, R$  “on each side” of  $\phi$  as in Proposition 23. Let  $c_1, c_2 \in [0, 1)$ , with  $c_2 \geq c_1$ . Suppose that  $\phi_1 : [0, 1] \rightarrow Q$  is a curve with  $\phi_1(0) = \phi(c_1)$   $\phi_1(1) = \phi(c_2)$ ,  $\phi_1([0, 1])$  is disjoint from  $\phi([0, 1])$  and the curve  $\phi_1([0, 0.5])$  approaches  $\phi([0, 1])$  from the left  $L$  and  $\phi_1([0.5, 1])$  approaches  $\phi([0, 1])$  from the right  $R$ .*

## 18:16 A Constant Factor Approximation for Directed Feedback Vertex Set

Then the curve  $\phi_2 : [0, 1] \rightarrow Q$ ,  $\phi_2(x) = \phi(c_2 - x)$  if  $x \leq c_2 - c_1$  and  $\phi_2(x) = \phi_1(\frac{1}{c_2 - c_1}(x - c_2 + c_1))$  for  $x > c_2 - c_1$ , that is the curve obtained by joining the portion of  $\phi$  from  $c_1$  to  $c_2$  to  $\phi_1([0, 1])$  does not divide  $Q$  into 2 regions.

**Proof.** We prove Proposition 23, Proposition 24, and Proposition 25. We use the following corollary of the tubular neighbourhood theorem (see for instance [4]).

► **Proposition 27** (corollary of tubular neighbourhood theorem [4]). *Given a curve  $C'$  embedded in a surface  $Q$  there is an open neighbourhood  $U$  of  $C'$  and an open set  $V$  in  $C' \times \mathbb{R}$  such that there is a diffeomorphism  $\phi : U \rightarrow V$  with  $\phi(C') = C' \times \{0\}$ .*

Let  $w : [0, 1] \rightarrow C'$  with  $w(0) = w(1)$ ,  $w(0.5) = c_2$  be a parameterization of  $C'$ . We define distance on  $C' \times \mathbb{R}$  by  $\text{dist}((a_1, b_1), (a_2, b_2)) := (\text{dist}_Q(a_1, a_2)^2 + |b_1 - b_2|^2)^{\frac{1}{2}}$ , where  $\text{dist}_Q(a_1, a_2)$  is the geodesic distance between  $a_1$  and  $a_2$  in  $Q$ . For each  $v \in C'$  let  $q_v$  be the minimum of 1 and  $\sup\{q' : B(v, q') \subset V\}$ .  $q_v$  is continuous in  $v$  and  $q_v > 0 \forall v \in C'$ . By compactness of  $C'$ ,  $q := \min_{v \in C'} q_v$  exists and is positive. Now define  $L' = \phi^{-1}(C' \times (-q, 0))$ ,  $R' = \phi^{-1}(C' \times (0, q))$ .

Define  $U' = L' \cup C' \cup R'$ . Let  $f : [0, 1] \rightarrow Q$  be a curve with  $f(x) \notin C'$  for any  $x \in [0, 1]$ . By continuity of  $f$  there is some  $\beta \in (0, 1)$  for which  $f((\beta, 1]) \subset U'$ . We claim  $f((\beta, 1]) \subset L'$  or  $f((\beta, 1]) \subset R'$ . If  $\phi(f((\beta, 1)))$  contains a point in  $C' \times (-\infty, 0)$  and a point in  $\phi^{-1}(C' \times (0, \infty))$  then by continuity  $\phi(f((\beta, 1)))$  contains a point in  $C' \times \{0\}$  and hence  $f((\beta, 1))$  contains a point in  $C'$  which is a contradiction. Thus, either  $\phi(f((\beta, 1))) \subset C' \times (-\infty, 0)$  or  $\phi(f((\beta, 1))) \subset C' \times (0, \infty)$ . If  $\phi(f((\beta, 1))) \subset C' \times (-\infty, 0)$ , then  $f((\beta, 1)) \subset L'$ . If  $\phi(f((\beta, 1))) \subset C' \times (0, \infty)$ , then  $f((\beta, 1)) \subset R'$ .

For each  $f_i$  there exists  $\beta_i$  for which  $f((\beta_i, 1]) \in L'$  or  $f((\beta_i, 1]) \in R'$ . For each  $x \in C'$  define  $r'_x$  to be the supremum of all radius  $r''_x$  for which the ball  $B(x, r''_x)$  of radius  $r''_x$  is entirely contained in  $U'$  and for which  $B(x, r''_x)$  is disjoint from  $f_1([0, \beta_1]), f_2([0, \beta_2]), f_3([0, \beta_3]), \dots, f_{l'}([0, \beta_{l'}]), h_1([0, 1]), h_2([0, 1]), \dots, h_{l'}([0, 1])$ . If the supremum does not exist, set  $r'_x = \infty$ . Define  $r_x = \min\{1, r'_x\}$ . Again  $r_x > 0$  for all  $x \in C'$  and is continuous in  $x$ . Since  $C'$  is a compact set,  $r := \min_{x \in C'} r_x$  exists and is positive. Note each curve  $f_i([0, 1])$  is disjoint from at least one of  $L' \cap \{B(x, r) : x \in C'\}, R' \cap \{B(x, r) : x \in C'\}$  and each curve  $h_i([0, 1])$  is disjoint from both  $L' \cap \{B(x, r) : x \in C'\}, R' \cap \{B(x, r) : x \in C'\}$ .

Let us show that by making  $r$  smaller if necessary  $B(v, r) \setminus C'$  contains two connected components.

► **Proposition 28.** *Given a (piece-wise smooth non-self-intersecting) curve  $f : [0, 1] \rightarrow \mathbb{R}^2$  with  $t_0 \in (0, 1)$  there exists  $r > 0$  for which  $B(f(t_0), r) \setminus f([0, 1])$  contains exactly two components.*

**Proof.** Let  $[t_0, t_1]$  be an interval in which  $f$  is smooth.

Let  $f(t) = f(t_0) + (t - t_0)\nabla f(t_0) + g(t - t_0)$ . By smoothness of  $f$ ,  $\nabla g(t - t_0)$  is bounded for  $t \in [t_0, t_1]$  and  $g(t - t_0) = o(t - t_0)$ . Differentiating  $\|f(t) - f(t_0)\|^2 = \|(t - t_0)\nabla f(t_0) + g(t - t_0)\|^2$  we obtain

$$\begin{aligned} \frac{d}{dt} \|f(t) - f(t_0)\|^2 &= 2(\nabla f(t_0) + \nabla g(t - t_0))^t ((t - t_0)\nabla f(t_0) + g(t - t_0)) \\ &= 2(\nabla f(t_0) + o(1))^t ((t - t_0)\nabla f(t_0) + o(t - t_0)) \\ &= 2(\nabla f(t_0) + o(1))^t (t - t_0)\nabla f(t_0) + o(t - t_0) \end{aligned}$$

For  $t$  close enough to  $t_0$  the last line is positive. This implies that for some  $t_2 > t_0$ ,  $\|f(t) - f(t_0)\|^2$  is increasing on  $[t_0, t_2]$ . Likewise, for some  $t_3 < t_0$ ,  $\|f(t) - f(t_0)\|^2$  is decreasing on  $[t_3, t_0]$ .

Let  $r > 0$  be such that  $\|f(t_0) - f(t)\| \geq 2r$  for all  $t \in [0, 1] \setminus \text{frac}([t_3, t_2])$ , where  $\text{frac}(x)$  is the fractional part of  $x$ . Then  $\|f(t_0) - f(t_2)\|, \|f(t_0) - f(t_3)\| \geq 2r$ . Then since  $\|f(t_0) - f(t)\|$  is increasing on  $[t_0, t_2]$  there is exactly one  $t_4 \in [t_0, t_2]$  with  $\|f(t_0) - f(t_4)\| = r$ . Likewise, there is exactly one  $t_5 \in [t_3, t_0]$  with  $\|f(t_0) - f(t_5)\| = r$ . So  $f([t_5, t_4])$  forms a simple curve in the closed ball  $\bar{B}(f(t_0), r)$  with endpoints on the boundary and  $f((t_5, t_4))$  lying in the interior. It follows from the Jordan curve theorem that  $B(f(t_0), r) \setminus f([0, 1]) = B(f(t_0), r) \setminus f((t_5, t_4))$  contains exactly two connected components.  $\blacktriangleleft$

Let  $v \in C'$ . For small enough  $r_0$ ,  $B(v, r_0)$  is diffeomorphic to the open disk  $B(0, r_0)$  in  $\mathbb{R}^2$  via some diffeomorphism  $\psi$  with  $\psi(v) = (0, 0)$ . Let  $w$  be a parameterization of  $C'$  with  $w(0.5) = v$ . Let  $t_1 := \inf\{t : w([t, 0.5]) \subset B(v, r_0)\}$  and  $t_2 := \sup\{t : w([0.5, t]) \subset B(v, r_0)\}$  that is  $[t_1, t_2]$  is a maximal interval for which  $w([t_1, t_2]) \subset B(v, r_0)$  by continuity  $t_1 < 0.5 < t_2$ . Choose  $r_1 > 0$  less than the distance from  $v$  to  $C' \setminus w((t_1, t_2))$  and  $r_1 < \text{dist}_Q(v, w(t_1)), \text{dist}_Q(v, w(t_2))$ . From the previous proposition there exists  $r_2 > 0$  such that  $B_{\mathbb{R}^2}(\psi(v), r_2) \setminus \psi(w([t_1, t_2]))$  contains exactly two connected components. By making  $r$  smaller than  $r_1$  and  $r_2$  if necessary we get that for any  $0 < \hat{r} \leq r$ ,  $B_{\mathbb{R}^2}(\psi(v), \hat{r}) \setminus \psi(C')$  contains exactly two connected components. Thus,  $B(v, \hat{r}) \setminus C'$  contains exactly two connected components.

Define  $L = L' \cap \{B(x, r) : x \in C'\}$ ,  $R = R' \cap \{B(x, r) : x \in C'\}$ . Each curve  $f : [0, 1] \rightarrow Q$  with  $f(x) \notin C'$  for any  $x \in [0, 1]$  satisfies  $f((\beta, 1)) \subset L$  or  $f((\beta, 1)) \subset R$ . Each curve  $f_i([0, 1])$  is disjoint from at least one of  $L, R$  and each curve  $h_i([0, 1])$  is disjoint from both  $L, R$ .

Since  $\phi(v) = \{v\} \times \{0\}$ ,  $\phi(B(v, \hat{r}))$  intersects both  $\{v\} \times (-\infty, 0)$  and  $\{v\} \times (0, \infty)$ . Recall  $B(v, \hat{r}) \subset L \cup R \cup C'$ , so  $B(v, \hat{r})$  intersects both  $L$  and  $R$ . Since there is no path from  $L$  to  $R$  in  $L \cup R \cup C'$  one component of  $B(v, \hat{r}) \setminus C'$  is contained in  $L$  and the other is contained in  $R$ .

For each  $v \in C'$  let  $y_v$  be the supremum of  $\{y'_v \geq 0 : \{v\} \times (-y'_v, y'_v) \subset \phi(B(v, r))\}$ . Again  $y_v$  is positive and continuous in  $v$  so  $y := \min_{v \in C'} y_v$  exists and is positive. Then  $C' \times (-y, 0) \subset \phi(L)$  and  $C' \times (0, y) \subset \phi(R)$ . Define the curves  $f_L, f_R$  to be parameterizations of  $\phi^{-1}(C' \times \{-\frac{y}{2}\})$  and  $\phi^{-1}(C' \times \{\frac{y}{2}\})$  respectively.

Lastly given a curve  $h : [0, 1] \rightarrow Q$  be any curve that reaches  $C'$  from the right at a point  $c_2 = h(1)$  on  $C'$ , leaves  $C'$  from the left at  $c_1 = h(0)$  and  $C'_{c_1, c_2}$  be a subcurve of  $C'$  with endpoints  $c_1$  and  $c_2$ . Let  $j : [0, 1] \rightarrow C'_{c_1, c_2}$  be a parameterization of  $C'_{c_1, c_2}$  and denote  $\bar{j} : [0, 2] \rightarrow Q$  by  $\bar{j}(t) = h(t)$  for  $t \in [0, 1]$  and  $\bar{j}(t) = j(t - 1)$  for  $t > 1$ . Informally speaking, we “slightly shift” all points in  $L \cup C' \cup R$  to the right while keeping  $h([0, 1]) \cap L$  to the left of  $C'$ . Let  $\phi(x) = (\phi_1(x), \phi_2(x))$ ,  $\phi^{-1}(x) = (\phi_1^{-1}(x), \phi_2^{-1}(x))$ .

Let  $\gamma : C' \rightarrow (-y, 0)$  be any continuous function such that  $\gamma(c_2) = 0$  and for any  $t \in [0, 1]$  for which  $\phi(h(t)) \in C' \times (-y, 0)$ ,  $(\phi(h(t))_2) < \gamma(\phi(h(t))_1) < 0$ . Informally  $\gamma$  is a curve lying to the right of  $\phi(h([0, 1])) \cap C' \times (-y, 0)$  and to the left of  $[0, 1] \times \{0\}$ . Such  $\gamma$  exists for instance define  $-\gamma(t)$  to be half of the minimum of the distance from the point  $(t, 0)$  to  $\phi(h([0, 1])) \cap C' \times (-y, 0)$  and  $y$ .

Define  $\bar{\gamma} : C' \times (-y, y) \rightarrow C' \times (-y, y)$  as follows. For  $(a, b) \in C' \times (-y, y)$  if  $b < \frac{\gamma(a)}{4}$  define  $\bar{\gamma}(a, b) = (a, b)$ . If  $\frac{\gamma(a)}{4} \leq b < 0$ , define  $\bar{\gamma}(a, b) = (a, \frac{\gamma(a)}{4} + 2(b - \frac{\gamma(a)}{4}))$ . If  $0 \leq b \leq \frac{-\gamma(a)}{2}$ , define  $\bar{\gamma}(a, b) = (a, \frac{-\gamma(a)}{4} + \frac{b}{2})$ . If  $b \geq \frac{-\gamma(a)}{2}$   $\bar{\gamma}(a, b) = (a, b)$ . Informally,  $\bar{\gamma}$  shifts  $C' \times (-y, y)$  to the right while keeping  $\phi(h([0, 1])) \cap C' \times (-y, 0)$  left of  $C' \times \{0\}$ .

Define  $\hat{\gamma} : Q \rightarrow Q$  by  $\hat{\gamma}(v) = v$  if  $v \notin \phi^{-1}(C' \times (-y, y))$  and  $\hat{\gamma}(v) = \phi^{-1}\bar{\gamma}(\phi(v))$  if  $v \in \phi^{-1}(C' \times (-y, y))$ . Note  $\hat{\gamma}$  is a homeomorphism from  $\phi^{-1}(C' \times (-y, y))$  to  $\phi^{-1}(C' \times (-y, y))$  and from  $Q \setminus \phi^{-1}(C' \times (-y, y))$  to  $Q \setminus \phi^{-1}(C' \times (-y, y))$ . Further,  $\hat{\gamma}$  agrees on the boundary of  $\phi^{-1}(C' \times (-y, y))$  and  $Q \setminus \phi^{-1}(C' \times (-y, y))$ , that is for sequence  $\{a_i\}_{i=1}^\infty$  converging to a boundary point  $a$  of  $\phi^{-1}(C' \times (-y, y))$  (resp  $Q \setminus \phi^{-1}(C' \times (-y, y))$ ) the sequence  $\{\hat{\gamma}(a_i)\}_{i=1}^\infty$  converges to  $\hat{\gamma}(a)$ . Thus,  $\hat{\gamma}$  is a homeomorphism on  $Q$ , in fact, it turns out to be a continuous deformation.

## 18:18 A Constant Factor Approximation for Directed Feedback Vertex Set

Define  $\hat{h} : [0, 2] \rightarrow Q$  by  $\hat{h}(t) = \hat{\gamma}(\bar{j}(t))$ . Then  $\hat{\gamma}$  is a homeomorphism mapping  $\hat{h}([0, 2])$  to  $\bar{j}([0, 2])$ .  $\blacktriangleleft$

The actual statement of the tubular neighbourhood theorem involves first defining the normal fibre  $N_x$  as the quotient  $T_x Q / T_x C$  where  $T_x Q$  and  $T_x C$  are the tangent plane and tangent to the curve  $C$  at  $x$  and the normal bundle  $NX$  as  $\{(x, v) : x \in C, v \in N_x\}$ .

► **Theorem 29** (tubular neighbourhood theorem). *There are open sets  $U$  in  $Q$  containing  $C$  and  $V$  in  $NX$  such that there is a diffeomorphism  $\gamma : U \rightarrow V$ .*

Let us quickly show how the version of the tubular neighbourhood theorem in Proposition 27 follows from the tubular neighbourhood theorem.

By orientability each point  $x \in Q$  has a normal vector  $n(x)$  and  $n$  is continuous. We may parameterize  $C'$  with a function  $\psi : [0, \beta] \rightarrow C'$  with derivative  $\psi'(x) = 1$  for some  $\beta$ . Define  $v(x)$  to be of unit norm and positively orthogonal to  $n(x)$ ,  $\phi'(x)$  that is  $n(x)^t v(x) = (\phi'(x))^t v(x) = 0$  and  $[n(x) \ \psi'(x) \ v(x)]$  has determinant 1. By the inverse function theorem,  $v(x)$  is continuous.  $n(x), \psi'(x), v(x)$  is a basis for  $\mathbb{R}^3$  known as the Darboux frame. Since  $v(x)$  is orthogonal to  $n(x)$  it lies in the tangent plane  $T_x Q$  since  $v(x)$  is orthogonal to  $\phi'(x)$ ,  $v(x), \phi'(x)$  is a basis for  $T_x Q$ . Thus,  $N_x = T_x Q / T_x C$  is diffeomorphic to  $\{av(x) : a \in \mathbb{R}\}$  which is diffeomorphic to  $\mathbb{R}$ . Thus,  $NX$  is diffeomorphic to  $C' \times \mathbb{R}$ .

To prove Proposition 26, we first prove the following special case.

► **Proposition 30.** *Let  $Q$  be an orientable surface and  $\phi : [0, 1] \rightarrow Q$  a closed curve not dividing  $Q$  into 2 regions with disjoint subsets  $L, R$  “on each side” of  $\phi$  as in Proposition 23. Let  $c \in [0, 1]$ , with  $c_2 \geq c_1$ . Suppose that  $\phi_1 : [0, 1] \rightarrow Q$  is a curve with  $\phi_1(0) = \phi(c)$ ,  $\phi_1(1) = \phi(c)$ ,  $\phi_1((0, 1))$  is disjoint from  $\phi([0, 1])$  and the curve  $\phi_1([0, 0.5])$  approaches  $\phi([0, 1])$  from the left  $L$  and  $\phi_1([0.5, 1])$  approaches  $\phi([0, 1])$  from the right  $R$ .*

*Then  $\phi_1$  does not divide  $Q$  into 2 regions.*

**Proof.** Suppose for a contradiction that  $\phi_1$  divides  $Q$  into 2 regions. It's a well-known result that one of the regions  $Q_1$  must be homeomorphic to an open disk. Let  $Q_2$  be the other region.

There exists a small radius  $r_1$  for which the ball  $B(\phi(c), r)$  of radius  $r_1$  about  $\phi(c)$  such that  $B(\phi(c), r_1)$  is homeomorphic to an open disk. Let  $r$  be as in Proposition 23 and define  $r_2 = \min\{r, r_1\}$ .

► **Definition 31.** *Given 2 curves  $f_1, f_2 : [0, 1] \rightarrow Q$  on a surface  $Q$ , with  $f_1(0.5) = f_2(0.5)$  and  $f_1(x) \neq f_2(y) \ \forall x, y \in ([0, 1] \setminus \{0.5\})$ , that is they intersect only at  $f_1(0.5)$ , we say that  $f_1$  crosses  $f_2$  at  $f_1(0.5)$  if there exists  $r_0 > 0$  such that for all  $r \leq r_0$  such that  $f_2$  intersects both regions of  $B(f_1(0.5), r) \setminus f_1([0, 1])$ , where  $B(p, r)$  is the open ball around  $p$  of radius  $r$ .*

► **Lemma 32.** *For two curves  $f_1, f_2$  on a surface  $Q$  and  $p$  a point on both curves,  $f_1$  crosses  $f_2$  at  $p$  if and only if  $f_2$  crosses  $f_1$  at  $p$ .*

**Proof.** Let  $f_1(b_1) = p = f_2(b_2)$ . Let  $L_1, R_1$  and  $r_1$  (resp  $L_2, R_2$  and  $r_2$ ) be the left, right and radius respectively for  $f_1$  (resp.  $f_2$ ) as guaranteed by Proposition 23. Define  $r = \min\{r_1, r_2\}$ .

▷ **Claim 33.**  $f_1$  crosses  $f_2$  at  $p$  if and only if for all  $r \geq r_0 > 0$  none of  $L_1 \cap B(p, r_0)$ ,  $R_1 \cap B(p, r_0)$ ,  $L_2 \cap B(p, r_0)$ ,  $R_2 \cap B(p, r_0)$  is contained in another.



Proof. Suppose that  $f_1$  crosses  $f_2$ . Let  $r \geq r_0 > 0$ . Let  $t_L, t_R \in \mathbb{R}$  be such that  $f_2(t_L) \in L_1 \cap B(p, r_0)$ ,  $f_2(t_R) \in R_1 \cap B(p, r_0)$ . Since  $R_1 \cap B(p, r_0), L_1 \cap B(p, r_0)$  are open, there exists  $r_L, r_R$  be such that  $B(f_2(t_L), r_L) \subset L_1 \cap B(p, r_0)$  and  $B(f_2(t_R), r_R) \subset R_1 \cap B(p, r_0)$ . Since  $B(f_2(t_L), r_L) \cap L_2, B(f_2(t_L), r_L) \cap R_2, B(f_2(t_R), r_R) \cap L_2, B(f_2(t_R), r_R) \cap R_2 \neq \emptyset$ , none of  $L_1 \cap B(p, r_0), R_1 \cap B(p, r_0), L_2 \cap B(p, r_0), R_2 \cap B(p, r_0)$  is contained in another.

Conversely, suppose that for any  $0 < r_0 \leq r$  none of  $L_1 \cap B(p, r_0), R_1 \cap B(p, r_0), L_2 \cap B(p, r_0), R_2 \cap B(p, r_0)$  is contained in another. Then for any  $0 < r_0 \leq r$ , if  $f_2$  does not intersect  $L_1 \cap B(p, r_0)$ , then  $L_1 \cap B(p, r_0)$  is connected in  $B(p, r_0) \setminus f_2$ , that is  $L_1 \cap B(p, r_0)$  is contained in one of the two components  $L_2 \cap B(p, r_0), R_2 \cap B(p, r_0)$  of  $B(p, r_0) \setminus f_2$ . Hence  $f_2$  intersects  $L_1 \cap B(p, r_0)$ , likewise  $f_2$  intersects  $R_1 \cap B(p, r_0)$ . Hence  $f_2$  crosses  $f_1$ .  $\triangleleft$

From the previous claim it's clear that crossing is a symmetric relation.  $\blacktriangleleft$

Define  $\phi_2(x) = \phi(\text{frac}(x - 0.5 + c))$ , where  $\text{frac}(x) = x - \lfloor x \rfloor$  is the fractional value of  $x$  and  $\phi_3(x) = \phi_1(\text{frac}(x - 0.5))$ , that is,  $\phi_2, \phi_3$  are reparameterized versions of  $\phi$  and  $\phi_1$ . For some  $0 < \beta_1 < \beta_2 < 1$   $\phi_1(x) \in L$  for  $x \in (0, \beta_1)$  and  $\phi_1(x) \in R$  for  $x \in (\beta_2, 1)$ .

Define  $\beta'_1 = \min\{\beta_1, 0.5\} + 0.5$   $\beta'_2 = \max\{\beta_2, 0.5\}$ . Then  $\phi_3((0.5, \beta'_1)) \subset L$  and  $\phi_3((\beta'_2, 0.5)) \subset R$ . Since  $L \cup R$  covers  $\bar{B}(\phi(c), r) \setminus \phi([0, 1])$ , this implies that  $\phi_3$  crosses  $\phi_2$ .

By Lemma 32  $\phi_2$  crosses  $\phi_3$ . Let  $L_{\phi_3}, R_{\phi_3}$  be the left and right of  $\phi_3$  as in Proposition 23. Since  $L_{\phi_3}, R_{\phi_3}$  are connected,  $L_{\phi_3}, R_{\phi_3}$  belong to different regions of  $Q \setminus \phi_3$ . Since  $\phi_2$  crosses  $\phi_3$ , there exists  $t_0$  for which  $\phi(t_0) \in Q_1$  and  $t_1$  for which  $\phi(t_1) \in Q_2$ . Let  $t_2 = \inf\{a \in [0, 1] : \exists b \in (a, 1] \text{ s.t. } \phi((a, b)) \subset Q_1\}$ ,  $t_3 = \sup\{b \in [t_2, 1] : \text{s.t. } \phi((t_2, b)) \subset Q_1\}$ , that is  $[t_2, t_3]$  is a maximal interval for which  $\phi([t_2, t_3]) \subset Q_1$ . It follows  $\phi(t_2), \phi(t_3)$  lie on the boundary of  $Q_1$ , that is on  $\phi_1$ . This implies  $t_2, t_3 \in \{0, 1\}$ . Since  $t_2 < t_3$   $t_2 = 0$  and  $t_3 = 1$ . This implies that  $\phi([0, 1])$  lies in  $Q_1 \cup \{\phi(0)\}$  contradicting that there exists  $t_1$  for which  $\phi(t_1) \in Q_2$ .  $\blacktriangleleft$

**Proof.** (of Proposition 26) Let  $f : [0, 1] \rightarrow C'$  be a parameterization of  $C'$  and let  $c = f^{-1}(c_1)$ . Note that by Proposition 25 the curve  $\phi_2$  is homeomorphic to a curve  $\phi_3$  that enters  $\phi$  from the right and leaves  $\phi$  from the left at the same point  $f(c)$ . By Proposition 30,  $\phi_3$  does not divide  $Q$  into 2 regions. Thus,  $\phi_2$  does not divide  $Q$  into 2 regions.  $\blacktriangleleft$

## References

- 1 M.A. Armstrong. *Basic Topology*. Undergraduate Texts in Mathematics. Springer New York, 2013. URL: <https://books.google.ca/books?id=NJbuBwAAQBAAJ>.
- 2 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.*, 12:289–297, 1999.
- 3 Piotr Berman and Grigory Yaroslavtsev. Primal-dual approximation algorithms for node-weighted network design in planar graphs. In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 50–60, Berlin, Heidelberg, 2012.
- 4 A.C. da Silva and LINK (Online service). *Lectures on Symplectic Geometry*. Number no. 1764 in Lecture Notes in Mathematics. Springer, 2001. URL: <https://books.google.ca/books?id=r9i6pXc7GEQC>.
- 5 E. D. Demaine and M. Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008.
- 6 Reinhard Diestel. *Graph Theory*. Springer Publishing Company, Incorporated, 5th edition, 2017.
- 7 G. Even, J. (Seffi) Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

- 8 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and eptas. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 748–759, Philadelphia, PA, USA, 2011. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=2133036.2133095>.
- 9 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. *CoRR*, 2016.
- 10 Georges Gardarin and Stefano Spaccapietra. Integrity of data bases: A general lockout algorithm with deadlock avoidance. In G. M. Nijssen, editor, *Modelling in Data Base Management Systems, Proceeding of the IFIP Working Conference on Modelling in Data Base Management Systems, Freudenstadt, Germany, January 5-8, 1976*, pages 395–412. North-Holland, 1976.
- 11 Michel X. Goemans and David P. Williamson. *The Primal-Dual Method for Approximation Algorithms and Its Application to Network Design Problems*, pages 144–191. PWS Publishing Co., USA, 1996.
- 12 Michel X. Goemans and David P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18(1):37–59, 1998.
- 13 Venkatesan Guruswami and Euiwoong Lee. Simple proof of hardness of feedback vertex set. *Theory of Computing*, 12(6):1–11, 2016. doi:10.4086/toc.2016.v012a006.
- 14 Allen Hatcher. Notes on basic 3-manifold topology, 2007. URL: <https://pi.math.cornell.edu/~hatcher/3M/3M.pdf>.
- 15 The Amplitwist (<https://mathoverflow.net/users/183188/the-amplitwist>). If a graph embedded on a surface is divided by a curve into a right and left that do not intersect can it be embedded on a surface of smaller genus? MathOverflow. (version: 2023-11-19). URL: <https://mathoverflow.net/q/458733>.
- 16 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2\_9.
- 17 L.C. Kinsey. *Topology of Surfaces*. Undergraduate Texts in Mathematics. Springer New York, 1997. URL: <https://books.google.g1/books?id=AKghdMm5W-YC>.
- 18 Lap Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2011. doi:10.1017/CB09780511977152.
- 19 Charles E. Leiserson and James B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6(1):5–35, 1991. doi:10.1007/BF01759032.
- 20 Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '85, pages 97–107, New York, NY, USA, 1985. Association for Computing Machinery. doi:10.1145/318593.318622.
- 21 Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2-approximating feedback vertex set in tournaments. *ACM Trans. Algorithms*, 17(2), April 2021. doi:10.1145/3446969.
- 22 J. Rydholm. Classification of compact orientable surfaces using morse theory, 2016.
- 23 Niklas Schlomberg, Hanjo Thiele, and Jens Vygen. Packing cycles in planar and bounded-genus graphs, 2023. arXiv:2207.00450.
- 24 Anke van Zuylen. Linear programming based approximation algorithms for feedback set problems in bipartite tournaments. In *TAMC*, 2009.
- 25 J. W. T. Youngs. Minimal imbeddings and the genus of a graph. *J. Math. Mech.*, 12:303–315, 1963.

# More Basis Reduction for Linear Codes: Backward Reduction, BKZ, Slide Reduction, and More

Surendra Ghentiyala ✉

Cornell University, Ithaca, NY, USA

Noah Stephens-Davidowitz ✉

Cornell University, Ithaca, NY, USA

---

## Abstract

---

We expand on recent exciting work of Debris-Alazard, Ducas, and van Woerden [Transactions on Information Theory, 2022], which introduced the notion of *basis reduction for codes*, in analogy with the extremely successful paradigm of basis reduction for lattices. We generalize DDvW’s LLL algorithm and size-reduction algorithm from codes over  $\mathbb{F}_2$  to codes over  $\mathbb{F}_q$ , and we further develop the theory of proper bases. We then show how to instantiate for codes the BKZ and slide-reduction algorithms, which are the two most important generalizations of the LLL algorithm for lattices.

Perhaps most importantly, we show a new and very efficient basis-reduction algorithm for codes, called *full backward reduction*. This algorithm is quite specific to codes and seems to have no analogue in the lattice setting. We prove that this algorithm finds vectors as short as LLL does in the worst case (i.e., within the Griesmer bound) and does so in less time. We also provide both heuristic and empirical evidence that it outperforms LLL in practice, and we give a variant of the algorithm that provably outperforms LLL (in some sense) for random codes.

Finally, we explore the promise and limitations of basis reduction for codes. In particular, we show upper and lower bounds on how “good” of a basis a code can have, and we show two additional illustrative algorithms that demonstrate some of the promise and the limitations of basis reduction for codes.

**2012 ACM Subject Classification** Theory of computation → Error-correcting codes

**Keywords and phrases** Linear Codes, Basis Reduction

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.19

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2408.08507> [21]

**Funding** *Surendra Ghentiyala*: This work is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

*Noah Stephens-Davidowitz*: This work is supported in part by the NSF under Grants Nos. CCF-2122230 and CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google. Some of this work was completed while the author was visiting the National University of Singapore and the Centre for Quantum Technologies

## 1 Introduction

### 1.1 Codes and lattices

A *linear code*  $\mathcal{C} \subseteq \mathbb{F}_q^n$  is a subspace of the vector space  $\mathbb{F}_q^n$  over the finite field  $\mathbb{F}_q$ , i.e., it is the set of all  $\mathbb{F}_q$ -linear combinations of a linearly independent list of vectors  $\mathbf{B} := (\mathbf{b}_1; \dots; \mathbf{b}_k)$ ,

$$\mathcal{C} := \mathcal{C}(\mathbf{B}) := \{z_1 \mathbf{b}_1 + \dots + z_k \mathbf{b}_k : z_i \in \mathbb{F}_q\}.$$

We call  $\mathbf{b}_1, \dots, \mathbf{b}_k$  a *basis* for the code and  $k$  the *dimension* of the code. We are interested in the geometry of the code induced by the Hamming weight  $|\mathbf{c}|$  for  $\mathbf{c} \in \mathbb{F}_q^n$ , which is simply the number of coordinates of  $\mathbf{c}$  that are non-zero. For example, it is natural to ask about a



© Surendra Ghentiyala and Noah Stephens-Davidowitz;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 19; pp. 19:1–19:22



Leibniz International Proceedings in Informatics

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

## 19:2 More Basis Reduction for Linear Codes

code's *minimum distance*, which is the minimal Hamming weight of a non-zero codeword, i.e.,

$$d_{\min}(\mathcal{C}) := \min_{\mathbf{c} \in \mathcal{C}_{\neq \mathbf{0}}} |\mathbf{c}|.$$

At a high level, there are two fundamental computational problems associated with linear codes. In the first, the goal is to find a short non-zero codeword – i.e., given a basis for a code  $\mathcal{C}$ , the goal is to find a non-zero codeword  $\mathbf{c} \in \mathcal{C}_{\neq \mathbf{0}}$  with relatively small Hamming weight  $|\mathbf{c}|$ . In the second, the goal is to find a codeword that is close to some target vector  $\mathbf{t} \in \mathbb{F}_q^n$  – i.e., given a basis for a code  $\mathcal{C}$  and a target vector  $\mathbf{t} \in \mathbb{F}_q^n$ , the goal is to find a codeword  $\mathbf{c} \in \mathcal{C}$  such that  $|\mathbf{c} - \mathbf{t}|$  is relatively small. (Here, we are being deliberately vague about what we mean by “relatively small.”)

We now describe another class of mathematical objects, which are also ubiquitous in computer science. Notice the striking similarity between the two descriptions.

A *lattice*  $\mathcal{L} \subset \mathbb{Q}^n$  is the set of all *integer* linear combinations of linearly independent basis vectors  $\mathbf{A} := (\mathbf{a}_1; \dots; \mathbf{a}_k) \in \mathbb{Q}^n$ , i.e.,

$$\mathcal{L} := \mathcal{L}(\mathbf{A}) := \{z_1 \mathbf{a}_1 + \dots + z_k \mathbf{a}_k : z_i \in \mathbb{Z}\}.$$

We are interested in the geometry of the lattice induced by the Euclidean norm  $\|\mathbf{a}\| := (a_1^2 + \dots + a_n^2)^{1/2}$ . In particular, it is natural to ask about a lattice's *minimum distance*, which is simply the minimal Euclidean norm of a non-zero lattice vector, i.e.,

$$\lambda_1(\mathcal{L}) := \min_{\mathbf{y} \in \mathcal{L}_{\neq \mathbf{0}}} \|\mathbf{y}\|.$$

At a high level, there are two fundamental computational problems associated with lattices. In the first, the goal is to find a short non-zero lattice vector – i.e., given a basis for a lattice  $\mathcal{L}$ , the goal is to find a non-zero lattice vector  $\mathbf{y} \in \mathcal{L}_{\neq \mathbf{0}}$  with relatively small Euclidean norm  $\|\mathbf{y}\|$ . In the second, the goal is to find a lattice vector that is close to some target vector  $\mathbf{t} \in \mathbb{Q}^n$  – i.e., given a basis for a lattice  $\mathcal{L}$  and a target vector  $\mathbf{t} \in \mathbb{Z}^n$ , the goal is to find a lattice vector  $\mathbf{y} \in \mathcal{L}$  such that  $\|\mathbf{y} - \mathbf{t}\|$  is relatively small. (Again, we are being deliberately vague here.)

Clearly, there is a strong analogy between linear codes and lattices, where to move from codes to lattices, one more-or-less just replaces a finite field  $\mathbb{F}_q$  with the integers  $\mathbb{Z}$  and the Hamming weight  $|\cdot|$  with the Euclidean norm  $\|\cdot\|$ . It is therefore no surprise that this analogy extends to many applications. For example, lattices and codes are both used for decoding noisy channels. They are both used for cryptography (see, e.g., [26, 3, 5, 23, 7, 30]; in fact, both are used specifically for *post-quantum* cryptography). And, many complexity-theoretic hardness results were proven simultaneously or nearly simultaneously for coding problems and for lattice problems, often with similar or even identical techniques.<sup>1</sup>

### 1.2 Basis reduction for lattices

However, the analogy between lattices and codes has been much less fruitful for algorithms. Of course, there are many algorithmic techniques for finding short or close codewords and many algorithmic techniques for finding short or close lattice vectors. But, in many parameter regimes of interest, the best algorithms for lattices are quite different from the best algorithms for codes.

---

<sup>1</sup> For example, similar results that came in separate works in the two settings include [15] and [14], [4] and [33], [27] and [19], [13, 2] and [32], etc. Works that simultaneously published the same results in the two settings include [9] and [18].

In the present work, we are interested in *basis reduction*, a ubiquitous algorithmic framework in the lattice literature. At a very high level, given a basis  $\mathbf{A} := (\mathbf{a}_1; \dots; \mathbf{a}_k)$  for a lattice  $\mathcal{L}$ , basis reduction algorithms work by attempting to find a “good” basis of  $\mathcal{L}$  (and in particular, a basis whose first vector  $\mathbf{a}_1$  is short) by repeatedly making “local changes” to the basis. Specifically, such algorithms manipulate the Gram-Schmidt vectors  $\tilde{\mathbf{a}}_1 := \mathbf{a}_1, \tilde{\mathbf{a}}_2 := \pi_{\{\mathbf{a}_1\}}^\perp(\mathbf{a}_2), \dots, \tilde{\mathbf{a}}_k := \pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{k-1}\}}^\perp(\mathbf{a}_k)$ . Here,  $\pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{i-1}\}}^\perp$  represents orthogonal projection onto the subspace orthogonal to  $\mathbf{a}_1, \dots, \mathbf{a}_{i-1}$ . Notice that we can view the Gram-Schmidt vector  $\tilde{\mathbf{a}}_i$  as a lattice vector in the lower-dimensional lattice generated by the *projected block*  $\mathbf{A}_{[i,j]} := \pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{i-1}\}}^\perp(\mathbf{a}_i; \dots; \mathbf{a}_j)$ . Basis reduction algorithms work by making many “local changes” to  $\mathbf{A}$ , i.e., changes to the block  $\mathbf{A}_{[i,j]}$  that leave the lattice  $\mathcal{L}(\mathbf{A}_{[i,j]})$  unchanged. The goal is to use such local changes to make earlier Gram-Schmidt vectors shorter. (In particular,  $\tilde{\mathbf{a}}_1 = \mathbf{a}_1$  is a non-zero lattice vector. So, if we can make the first Gram-Schmidt vector short, then we will have found a short non-zero lattice vector.) One accomplishes this, e.g., by finding a short non-zero vector  $\mathbf{y}$  in  $\mathcal{L}(\mathbf{A}_{[i,j]})$  and essentially replacing the first vector in the block with this vector  $\mathbf{y}$ . (Here, we are ignoring how exactly one does this replacement.)

This paradigm was introduced in the celebrated work of Lenstra, Lenstra, and Lovász [25], which described the polynomial-time LLL algorithm. Specifically, (ignoring important details that are not relevant to the present work) the LLL algorithm works by repeatedly replacing Gram-Schmidt vectors  $\tilde{\mathbf{a}}_i$  with a shortest non-zero vector in the lattice generated by the dimension-two block  $\mathbf{A}_{[i,i+1]}$ . The LLL algorithm itself has innumerable applications. (See, e.g., [29].) Furthermore, generalizations of LLL yield the most efficient algorithms for finding short non-zero lattice vectors in a wide range of parameter regimes, including those relevant to cryptography.

Specifically, the Block-Korkine-Zolotarev basis-reduction algorithm (BKZ), originally due to Schnorr [31], is a generalization of the LLL algorithm that works with larger blocks. It works by repeatedly modifying blocks  $\mathbf{A}_{[i,i+\beta-1]}$  of a lattice basis  $\mathbf{A} := (\mathbf{a}_1; \dots; \mathbf{a}_k)$  in order to ensure that the Gram-Schmidt vector  $\tilde{\mathbf{a}}_i$  is a shortest non-zero vector in the lattice generated by the block. Here, the parameter  $\beta \geq 2$  is called the *block size*, and the case  $\beta = 2$  corresponds to the LLL algorithm (ignoring some technical details). Larger block size  $\beta$  yields better bases consisting of shorter lattice vectors. But, to run the algorithm with block size  $\beta$ , we must find shortest non-zero vectors in a  $\beta$ -dimensional lattice, which requires running time  $2^{O(\beta)}$  with the best-known algorithms [6, 1, 12]. So, BKZ yields a tradeoff between the quality of the output basis and the running time of the algorithm. (Alternatively, one can view BKZ as a reduction from an approximate lattice problem in high dimensions to an exact lattice problem in lower dimensions, with the approximation factor depending on how much lower the resulting dimension is.)

BKZ is the fastest known algorithm in practice for the problems relevant to cryptography. However, BKZ is notoriously difficult to understand. Indeed, we still do not have a proof that the BKZ algorithm makes at most polynomially many calls to its  $\beta$ -dimensional oracle, nor do we have a tight bound on the quality of the bases output by BKZ, despite much effort. (See, e.g., [34]. However, for both the running time and the output quality of the basis, we now have a very good *heuristic* understanding [16, 11].)

Gama and Nguyen’s slide-reduction algorithm is an elegant alternative to BKZ that is far easier to analyze [20]. In particular, it outputs a basis whose quality (e.g., the length of the first vector) essentially matches our heuristic understanding of the behavior of BKZ, and it provably does so with polynomially many calls to a  $\beta$ -dimensional oracle for finding a shortest non-zero lattice vector. Indeed, for a wide range of parameters (including those relevant to cryptography), [20] yields the fastest algorithm with proven correctness for finding short non-zero lattice vectors.

### Dual reduction, and some foreshadowing

One of the key ideas used in Gama and Nguyen’s slide-reduction algorithm (as well as in other work, such as [28]) is the notion of a *dual-reduced block*  $\mathbf{A}_{[i,j]}$ . The motivation behind dual-reduced blocks starts with the observation that the product  $\|\tilde{\mathbf{a}}_i\| \cdots \|\tilde{\mathbf{a}}_j\|$  does not change if the lattice  $\mathcal{L}(\mathbf{A}_{[i,j]})$  is not changed. Formally, this quantity is the determinant of the lattice  $\mathcal{L}(\mathbf{A}_{[i,j]})$ , which is a lattice invariant. So, while it is perhaps more natural to think of basis reduction in terms of making earlier Gram-Schmidt vectors in a block shorter, with the ultimate goal of making  $\mathbf{a}_1$  short, one can more-or-less equivalently think of basis reduction in terms of making *later* Gram-Schmidt vectors *longer*.

One therefore defines a *dual-reduced* block as a block  $\mathbf{A}_{[i,j]}$  such that the last Gram-Schmidt vector  $\tilde{\mathbf{a}}_j$  is as *long* as it can be without changing the associated lattice  $\mathcal{L}(\mathbf{A}_{[i,j]})$ . When  $\beta := j - i + 1 > 2$ , a dual-reduced block is not the same as a block whose first Gram-Schmidt vector is as short as possible. However, there is still some pleasing symmetry here. In particular, it is not hard to show that the last Gram-Schmidt vector  $\tilde{\mathbf{a}}_j$  corresponds precisely to a non-zero (primitive) vector in the *dual* lattice of  $\mathcal{L}(\mathbf{A}_{[i,j]})$  with length  $1/\|\tilde{\mathbf{a}}_j\|$ . This of course explains the terminology. It also means that making the last Gram-Schmidt vector  $\tilde{\mathbf{a}}_j$  as long as possible corresponds to finding a shortest non-zero vector in the dual of  $\mathcal{L}(\mathbf{A}_{[i,j]})$ , while making the first Gram-Schmidt vector  $\tilde{\mathbf{a}}_i$  as short as possible of course corresponds to finding a shortest non-zero vector in  $\mathcal{L}(\mathbf{A}_{[i,j]})$  itself. Either way, this amounts to finding a shortest non-zero vector in a  $\beta$ -dimensional lattice, which takes time that is exponential in the block size  $\beta$ .

## 1.3 Basis reduction for codes!

As far as the authors are aware, until very recently there was no work attempting to use the ideas from basis reduction in the setting of linear codes. This changed with the recent exciting work of Debris-Alazard, Ducas, and van Woerden, who in particular showed a simple and elegant analogue of the LLL algorithm for codes [17].

Debris-Alazard, Ducas, and van Woerden provide a “dictionary” ([17, Table 1]) for translating important concepts in basis reduction from the setting of lattices to the setting of codes, and it is the starting point of our work. Below, we describe some of the dictionary from [17], as well as some of the barriers that one encounters when attempting to make basis reduction work for codes.

### 1.3.1 Projection, epipodal vectors, and proper bases

Recall that when one performs basis reduction on lattices, one works with the Gram-Schmidt vectors  $\tilde{\mathbf{a}}_i := \pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{i-1}\}}^\perp(\mathbf{a}_i)$  and the projected blocks  $\mathbf{A}_{[i,j]} := \pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{i-1}\}}^\perp(\mathbf{a}_i; \dots; \mathbf{a}_j)$ , i.e., the *orthogonal projection* of  $\mathbf{a}_i, \dots, \mathbf{a}_j$  onto the subspace  $\{\mathbf{a}_1, \dots, \mathbf{a}_{i-1}\}^\perp$  orthogonal to  $\mathbf{a}_1, \dots, \mathbf{a}_{i-1}$ .

So, if we wish to adapt basis reduction to the setting of linear codes (and we do!), it is natural to first ask what the analogue of *projection* is in the setting of codes. [17] gave a very nice answer to this question.<sup>2</sup> In particular, for a vector  $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}_q^n$  we call the set of indices  $i$  such that  $x_i$  is non-zero the *support* of  $\mathbf{x}$ , i.e.,

<sup>2</sup> [17] formally worked with the case  $q = 2$  everywhere. Rather than specialize our discussion here to  $\mathbb{F}_2$ , we will largely ignore this distinction in this part of the introduction. While the more general definitions that we provide here for arbitrary  $\mathbb{F}_q$  are new to the present work, when generalizing to  $\mathbb{F}_q$  is straightforward, we will not emphasize this in the introduction.

$$\text{Supp}(\mathbf{x}) := \{i \in [n] : x_i \neq 0\}.$$

Then, [17] define  $\mathbf{z} := \pi_{\{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}}^\perp(\mathbf{y})$  as follows. If  $i \in \bigcup_j \text{Supp}(\mathbf{x}_j)$ , then  $z_i = 0$ . Otherwise,  $z_i = y_i$ . In other words, the projection simply “zeros out the coordinates in the supports of the  $\mathbf{x}_j$ .” This notion of projection shares many (but certainly not all) of the features of orthogonal projection in  $\mathbb{R}^n$ , e.g., it is a linear contraction mapping that is idempotent.

Armed with this notion of projection, [17] then defined the *epipodal vectors* associated with a basis  $\mathbf{b}_1, \dots, \mathbf{b}_n$  as  $\mathbf{b}_1^+ := \mathbf{b}_1, \mathbf{b}_2^+ := \pi_{\{\mathbf{b}_1\}}^\perp(\mathbf{b}_2), \dots, \mathbf{b}_n^+ := \pi_{\{\mathbf{b}_1, \dots, \mathbf{b}_{n-1}\}}^\perp(\mathbf{b}_n)$ , in analogy with the Gram-Schmidt vectors. In this work, we go a bit further and define

$$\mathbf{B}_{[i,j]} := \pi_{\{\mathbf{b}_1, \dots, \mathbf{b}_{i-1}\}}^\perp(\mathbf{b}_i; \dots; \mathbf{b}_j),$$

in analogy with the notation in the literature on *lattice* basis reduction.

Here, [17] already encounters a bit of a roadblock. Namely, the epipodal vectors  $\mathbf{b}_i^+$  can be zero! E.g., if  $\mathbf{b}_1 = (1, 1, \dots, 1)$  is the all-ones vector, then  $\mathbf{b}_i^+$  will be zero for all  $i > 1$ <sup>3</sup>. This is rather troublesome and could lead to many issues down the road. For example, we might even encounter entire blocks  $\mathbf{B}_{[i,j]}$  that are zero! Fortunately, [17] shows how to get around this issue by defining *proper* bases, which are simply bases for which all the epipodal vectors are non-zero. They then observe that proper bases exist and are easy to compute. (In Section 4, we further develop the theory of proper bases.) So, this particular roadblock is manageable, but it already illustrates that the analogy between projection over  $\mathbb{F}_q^n$  and projection over  $\mathbb{R}^n$  is rather brittle.

The LLL algorithm for codes then follows elegantly from these definitions. In particular, a basis  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k)$  is *LLL-reduced* if it is proper and if  $\mathbf{b}_i^+$  is a shortest non-zero codeword in the dimension-two code generated by  $\mathbf{B}_{[i,i+1]}$  for all  $i = 1, \dots, k-1$ . [17] then show a simple algorithm that computes an LLL-reduced basis in polynomial time. Specifically, the algorithm repeatedly makes simple local changes to any block  $\mathbf{B}_{[i,i+1]}$  for which this condition is not satisfied until the basis is reduced.

In some ways, this new coding-theoretic algorithm is even more natural and elegant than the original LLL algorithm for lattices. For example, the original LLL algorithm had to worry about numerical blowup of the basis entries. And, the original LLL algorithm seems to require an additional slack factor  $\delta$  in order to avoid the situation in which the algorithm makes a large number of minuscule changes to the basis. Both of these issues do not arise over finite fields, where all vectors considered by the algorithm have entries in  $\mathbb{F}_q$  and integer lengths between 1 and  $n$ .

### 1.3.2 What’s a good basis and what is it good for?

Given the incredible importance of the LLL algorithm for lattices, it is a major achievement just to show that one can make sense of the notion of “LLL for codes.” But, once [17] have defined an LLL-reduced basis for codes and shown how to compute one efficiently, an obvious next question emerges: what can one do with such a basis?

In the case of lattices, the LLL algorithm is useful for many things, but primarily for the two most important computational lattice problems: finding short non-zero lattice vectors and finding close lattice vectors to a target. In particular, the first vector  $\mathbf{a}_1$  of an LLL-reduced basis is guaranteed to  $\|\mathbf{a}_1\| \leq 2^{k-1} \lambda_1(\mathcal{L})$ . This has proven to be incredibly useful, despite the apparently large approximation factor.

<sup>3</sup> Of course, similar issues do not occur over  $\mathbb{R}^n$ , because if  $\mathbf{a}_1, \dots, \mathbf{a}_k \in \mathbb{R}^n$  are linearly independent, then  $\pi_{\{\mathbf{a}_1, \dots, \mathbf{a}_{k-1}\}}^\perp(\mathbf{a}_k)$  cannot be zero.

For codes over  $\mathbb{F}_2$ , [17] show that the same is true, namely that if  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k)$  is an LLL-reduced basis for  $\mathcal{C} \subseteq \mathbb{F}_2^n$ , then  $|\mathbf{b}_1| \leq 2^{k-1}d_{\min}(\mathcal{C})$ . They prove this by showing that if  $\mathbf{b}_i^+$  has minimal length among the non-zero codewords in  $\mathcal{C}(\mathbf{B}_{[i,i+1]})$ , then  $|\mathbf{b}_i^+| \leq 2|\mathbf{b}_{i+1}^+|$ . It follows in particular that  $|\mathbf{b}_1| = |\mathbf{b}_1^+| \leq 2^{i-1}|\mathbf{b}_i^+|$  for all  $i$ . One can easily see that  $d_{\min}(\mathcal{C}) \geq \min_i |\mathbf{b}_i^+|$ , from which one immediately concludes that  $|\mathbf{b}_1^+| \leq 2^{k-1}d_{\min}(\mathcal{C})$ . A simple generalization of this argument shows that over  $\mathbb{F}_q$ , we have  $|\mathbf{b}_1^+| \leq q^{k-1}d_{\min}(\mathcal{C})$ . (We prove something more general and slightly stronger in the full version [21].)

However, notice that all codewords have length at most  $n$  and  $d_{\min}(\mathcal{C})$  is always at least 1. Therefore, an approximation factor of  $q^{k-1}$  is non-trivial only if  $n > q^{k-1}$ . Otherwise, literally any non-zero codeword has length less than  $q^{k-1}d_{\min}(\mathcal{C})$ ! On the other hand, if  $n > q^{k-1}$ , then we can anyway find an exact shortest vector in time roughly  $q^k n \lesssim n^2$  by simply enumerating all codewords. (The typical parameter regime of interest is when  $n = O(k)$ .)

In some sense, the issue here is that the space  $\mathbb{F}_q^n$  that codes live in is too “cramped.” While lattices are infinite sets that live in a space  $\mathbb{Q}^n$  with arbitrarily long and arbitrarily short non-zero vectors, codes are finite sets that live in a space  $\mathbb{F}_q^n$  in which all non-zero vectors have integer lengths between 1 and  $n$ . So, while for lattices, any approximation factor between one and, say,  $2^k$  is very interesting, for codes the region of interest is simply more narrow.

[17] go on to observe that because  $|\mathbf{b}_{i+1}^+|$  is an integer, for codes over  $\mathbb{F}_2$  an LLL-reduced basis must actually satisfy

$$|\mathbf{b}_{i+1}^+| \geq \left\lceil \frac{|\mathbf{b}_i^+|}{2} \right\rceil .$$

With this slightly stronger inequality together with the fact that  $\sum |\mathbf{b}_i^+| \leq n$ , they are able to show that  $\mathbf{b}_1$  of an LLL-reduced basis will meet the Griesmer bound [22],

$$\sum_{i=1}^k \left\lceil \frac{|\mathbf{b}_1|}{2^{i-1}} \right\rceil \leq n , \tag{1}$$

which is non-trivial. E.g., as long as  $k \geq \log n$ , it follows that

$$|\mathbf{b}_1| - \frac{\lceil \log |\mathbf{b}_1| \rceil}{2} \leq \frac{n - k}{2} + 1 \tag{2}$$

(We generalize this in the full version [21] to show that the appropriate generalization of LLL-reduced bases to codes over  $\mathbb{F}_q$  also meet the  $q$ -ary Griesmer bound.)

### 1.3.2.1 Finding close codewords and size reduction

For lattices, Babai also showed how to use an LLL-reduced basis to efficiently find *close* lattice vectors to a given target vector [10], and like the LLL algorithm itself, Babai’s algorithm too has innumerable applications. More generally, Babai’s algorithm tends to obtain closer lattice vectors if given a “better” basis, in a certain precise sense. [17] showed an analogous “size-reduction” algorithm that finds close codewords to a given target vector, with better performance given a “better” basis. Here, the notion of “better” is a bit subtle, but essentially a basis is “better” if the epipodal vectors tend to have similar lengths. (Notice that  $\sum_i |\mathbf{b}_i^+| = |\text{Supp}(\mathcal{C})|$ , so we cannot hope for all of the epipodal vectors to be short.)

The resulting size-reduction algorithm finds relatively close codewords remarkably quickly. (Indeed, in nearly linear time.) Furthermore, [17] showed how to use their size-reduction algorithm combined with techniques from information set decoding to speed up some



information set decoding algorithms for finding short codewords or close codewords to a target, without significantly affecting the quality of the output. For this, their key observation was the fact that typically most epipodal vectors actually have length one (particularly the later epipodal vectors, as one would expect given that later epipodal vectors by definition have more coordinates “zeroed out” by projection orthogonal to the earlier basis vectors) and that their size-reduction algorithm derives most of its advantage from how it treats the epipodal vectors with length greater than one. They therefore essentially run information set decoding on the code projected onto the support of the epipodal vectors with length one and then “lift” the result to a close codeword using their size-reduction algorithm.

They call the resulting algorithm Lee-Brickell-Babai because it is a hybrid of Babai-style size reduction and the Lee-Brickell algorithm [24]. The running time of this hybrid algorithm is dominated by the cost of running information set decoding on a code with dimension  $k - k_1$ , where

$$k_1 := |\{i : |\mathbf{b}_i^+| > 1\}|$$

is the number of epipodal vectors that do not have length 1. Indeed, the (heuristic) running time of this algorithm is better than Lee-Brickell by a factor that is exponential in  $k_1$ , so that even a small difference in  $k_1$  can make a large difference in the running time. They then show that LLL-reduced bases have  $k_1 \gtrsim \log n$  (for random codes) and show that this reduction in dimension can offer significant savings in the running time of information set decoding.

Indeed, though the details are not yet public, the current record in the coding problem challenges [8] was obtained by Ducas and Stevens, apparently using such techniques.

## 1.4 Our contribution

In this work, we continue the study of basis reduction for codes, expanding on and generalizing the results of [17] in many ways, and beginning to uncover a rich landscape of algorithms.

### 1.4.1 Expanding on the work of [17]

#### 1.4.1.1 Generalization to $\mathbb{F}_q$

Our first set of (perhaps relatively minor) contributions are generalizations of many of the ideas in [17] from  $\mathbb{F}_2$  to  $\mathbb{F}_q$ , a direction proposed in that work. In fact, they quite accurately anticipated this direction. So, we quote directly from [17, Section 1.3]:

In principle, the definitions, theorems and algorithms of this article should be generalizable to codes over  $\mathbb{F}_q$  endowed with the Hamming metric. . . Some algorithms may see their complexity grow by a factor  $\Theta(q)$ , meaning that the algorithms remains polynomial-time only for  $q = n^{O(1)}$ . It is natural to hope that such a generalised LLL would still match [the] Griesmer bound for  $q > 2$ . However, we expect that the analysis of the fundamental domain [which is necessary for understanding size reduction]. . . would become significantly harder to carry out.

In Section 3, we generalize from  $\mathbb{F}_2$  to  $\mathbb{F}_q$  the definitions of projection, epipodal vectors, and proper bases; the definition of an LLL-reduced bases and the LLL algorithm;<sup>4</sup> and the size-reduction algorithm and its associated fundamental domain. Some of this is admittedly

<sup>4</sup> We actually describe the LLL algorithm as a special case of the more general algorithms that we describe below. See the full version [21].

quite straightforward – e.g., given the definitions in [17] of projection, epipodal vectors, and proper bases for codes over  $\mathbb{F}_2$ , the corresponding definitions for codes over  $\mathbb{F}_q$  are immediate (and we have already presented them in this introduction). And, the definition of an LLL-reduced basis and of size reduction follow more-or-less immediately from these definitions. In particular, we do in fact confirm that LLL over  $\mathbb{F}_q$  achieves the Griesmer bound.

As [17] anticipated, the most difficult challenge that we encounter here is in the analysis of the fundamental domain that one obtains when one runs size reduction with a particular basis  $\mathbf{B}$ . We refer the reader to the full version [21] for the details.

(We do not encounter the running time issue described in the quote above – except for our algorithm computing the number of vectors of a given length in  $\mathcal{F}(\mathbf{B}^+)$ . In particular, our versions of the LLL algorithm and the size-reduction algorithm – and even our generalizations like slide reduction – run in time that is proportional to a small polynomial in  $\log q$ .)

Along the way, we make some modest improvements to the work of [17], even in the case of  $\mathbb{F}_2$ . In particular, using more careful analysis, we shave a factor of roughly  $n$  from the proven running time of LLL. (See the full version [21].)

#### 1.4.1.2 More on the theory of proper bases

In order to develop the basis-reduction algorithms that we will describe next, we found that it was first necessary to develop (in Section 4) some additional tools for understanding and working with *proper bases*, which might be of independent interest. Specifically, we define the concept of a *primitive codeword*, which is a non-zero codeword  $\mathbf{c}$  such that  $\text{Supp}(\mathbf{c})$  does not strictly contain the support of any other non-zero codeword. We then show that primitive codewords are closely related to proper bases. For example, we show that  $\mathbf{c}$  is the first vector in some proper basis if and only if  $\mathbf{c}$  is primitive, and that a basis is proper if and only if the epipodal vectors are primitive vectors in their respective projections.

We find this perspective to be quite useful for thinking about proper bases and basis reduction in general. In particular, we use this perspective to develop algorithms that perform basic operations on proper bases, such as inserting a primitive codeword into the first entry of a basis without affecting properness. The resulting algorithmic tools seem to be necessary for the larger-block-size versions of basis reduction that we describe below, in which our algorithms must make more complicated changes to a basis.

#### 1.4.2 Backward reduction and redundant sets

Our next contribution is the notion of *backward reduction*, described in Section 5. Recall that in the context of lattices, a key idea is the notion of a *dual-reduced* block  $\mathbf{A}_{[i,j]}$ , in which the *last* Gram-Schmidt vector  $\tilde{\mathbf{a}}_j$  is as *long* possible, while keeping  $\mathcal{L}(\mathbf{A}_{[i,j]})$  fixed.

Backward-reduced blocks are what we call the analogous notion for codes. Specifically, we say that a block  $\mathbf{B}_{[i,j]}$  is *backward reduced* if the last epipodal vector  $\mathbf{b}_j^+$  is as *long* as possible, while keeping  $\mathcal{C}(\mathbf{B}_{[i,j]})$  fixed. Just like in the case of lattices, this idea is motivated by an invariant. Here, the invariant is  $|\mathbf{b}_i^+| + \cdots + |\mathbf{b}_j^+|$ , which is precisely the support of the code  $\mathcal{C}(\mathbf{B}_{[i,j]})$ . So, if one wishes to make earlier epipodal vectors shorter (and we do!), then one will necessarily make later epipodal vectors longer, and vice versa. In particular, in the case of LLL, when the block size  $\beta := j - i + 1$  is equal to 2, there is no difference between minimizing the length of the first epipodal vector and maximizing the length of the second epipodal vector. So, if one wishes, one can describe the LLL algorithm in [17] as working by repeatedly backward reducing blocks  $\mathbf{B}_{[i,i+1]}$ .

The above definition of course leads naturally to two questions. First of all, how do we produce a backward-reduced block (for block size larger than 2)? And, second, what can we say about them? Specifically, what can we say about the length  $|\mathbf{b}_j^+|$  of the last epipodal vector in a backward-reduced block  $\mathbf{B}_{[i,j]}$ ?

One might get discouraged here, as one quickly discovers that long last epipodal vectors *do not* correspond to short non-zero codewords in the dual code. So, the beautiful duality that arises in the setting of lattices simply fails in our new context. (The *only* exception is that last epipodal vectors with length *exactly* two correspond to dual vectors with length *exactly* two.) This is why we use the terminology “backward reduced” rather than “dual reduced.” One might fear that the absence of this correspondence would make backward-reduced blocks very difficult to work with.

Instead, we show that long last epipodal vectors  $\mathbf{b}_j^+$  in a block  $\mathbf{B}_{[i,j]}$  have a simple interpretation. They correspond precisely to large *redundant sets of coordinates* of the code  $\mathcal{C}(\mathbf{B}_{[i,j]})$ . In the special case when  $q = 2$ , a redundant set  $S \subseteq [n]$  of coordinates is simply a set of coordinates in the support of the code such that for every  $a, b \in S$  and every codeword  $\mathbf{c}$ ,  $c_a = c_b$ . For larger  $q$ , we instead have the guarantee that  $c_a = z c_b$  for fixed non-zero scalar  $z \in \mathbb{F}_q^*$  depending only on  $a$  and  $b$ . In particular, maximal redundant sets correspond precisely to the non-zero coordinates in a last epipodal vector. (See Lemma 8.)

This characterization immediately yields an algorithm for backward reducing a block. (See Algorithm 2.) In fact, finding a backward-reduced block boils down to finding a set of most common elements in a list of at most  $n$  non-zero columns, each consisting of  $\beta := j - i + 1$  elements from  $\mathbb{F}_q$ . One quite surprising consequence of this is that one can actually find backward-reduced blocks efficiently, even for large  $\beta$ ! (Compare this to the case of lattices, where finding a dual-reduced block for large  $\beta$  is equivalent to the NP-hard problem of finding a shortest non-zero vector in a lattice of dimension  $\beta$ .)

Furthermore, this simple combinatorial characterization of backward-reduced blocks makes it quite easy to prove a simple tight lower bound on the length of  $\mathbf{b}_j^+$  in a backward-reduced block  $\mathbf{B}_{[i,j]}$ . (See the full version [21].) Indeed, such a proof follows immediately from the pigeonhole principle. This makes backward-reduced blocks quite easy to analyze. In contrast, as we will see below, *forward-reduced* blocks, in which the first epipodal vector  $\mathbf{b}_i^+$  is as short as possible, are rather difficult to analyze for  $\beta > 2$ .

### 1.4.3 Fully backward-reduced bases

With this new characterization of backward-reduced blocks and the realization that we can backward reduce a block quite efficiently, we go on to define the notion of a *fully backward-reduced basis*. We say that a basis is *fully backward reduced* if *all* of the prefixes  $\mathbf{B}_{[1,j]}$  are backward reduced for all  $1 \leq j \leq k$ .<sup>5</sup> In fact, we are slightly more general than this, and consider bases that satisfy this requirement for all  $j$  up to some threshold  $\tau \leq k$ .

We show that a fully backward-reduced basis achieves the Griesmer bound (Equation (1) for  $q = 2$ ), just like an LLL-reduced basis. This is actually unsurprising, since it is not difficult to see that when the threshold  $\tau = k$  is maximal, a fully backward-reduced basis is also LLL reduced. However, even when  $\tau = \log_q n$ , we still show that a backward-reduced basis achieves the Griesmer bound. (See Theorem 16.)

<sup>5</sup> Notice that this implies that  $\mathbf{B}_{[i,j]}$  is also backward reduced for any  $1 \leq i < j$ . So, such bases really are *fully* backward reduced.

We then show a very simple and very efficient algorithm for computing fully backward-reduced bases. In particular, if the algorithm is given as input a *proper* basis, then it will convert it into a fully backward-reduced basis up to threshold  $\tau$  in time  $O(\tau^2 n \text{polylog}(n, q))$ . Notice that this is *extremely* efficient when  $\tau \leq \text{poly}(\log_q n)$ .<sup>6</sup> Indeed, for most parameters of interest, this running time is in fact less than the time  $O(nk \log q)$  needed simply to read the input basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$ . (Of course, this is possible because the algorithm only looks at the first  $\tau$  rows of the input basis.) So, if one already has a proper basis, one can convert it into a fully backward-reduced basis nearly for free.<sup>7</sup>

In contrast, the LLL algorithm runs in time  $O(kn^2 \log^2 q)$ . One *can* perform a similar “threshold” trick and run the LLL algorithm only on the first  $\tau$  basis vectors for  $\tau = \lceil \log_q n \rceil$  (which would still imply that  $|\mathbf{b}_1|$  must be bounded by the Griesmer bound). But, this would still yield a running time of  $\Omega(\tau n^2 \log^2 q)$  in the worst case. The speedup that we achieve from fully backward reduction comes from the combination of this threshold trick together with the fact that fully backward reduction runs in time proportional to  $\tau^2 n$ , rather than  $\tau n^2$ .

Furthermore, we show empirically that the resulting algorithm tends to produce better bases than LLL in practice. (See the full version [21].)

(It seems unlikely that any similar algorithm exists for lattices for two reasons. First, in the setting of lattices, computing a dual-reduced basis for large block sizes is computationally infeasible. Second, while for codes it is not unreasonable to look for a short non-zero codeword in the subcode generated by the first  $\tau$  basis vectors, for lattices the lattice generated by the first  $k - 1$  basis vectors often contains no shorter non-zero vectors than the basis vectors themselves, even when the full lattice contains much shorter vectors.)

### 1.4.3.1 Heuristic analysis of full backward reduction

We also provide heuristic analysis of full backward reduction, providing a compelling heuristic explanation for why its performance in practice seems to be much better than what worst-case analysis suggests. In particular, recall that we essentially characterize the length of the last epipodal vector of a backward-reduced block  $\mathbf{B}_{[i,j]}$  in terms of the maximal number of times that a column in  $\mathbf{B}_{[i,j]}$  repeats. We then naturally use the pigeonhole principle to argue that for suitable parameters there must be a column that repeats many times.

E.g., for  $q = 2$ , there must be at least one repeated non-zero column if the number of non-zero columns  $s$  is larger than the number of possible non-zero columns  $2^\beta - 1$ , where  $\beta := j - i + 1$  is the length of a column. This analysis is of course tight in the worst case. However, in the average case, we know from the birthday paradox that we should expect to see a repeated column even if  $s$  is roughly  $2^{\beta/2}$ , rather than  $2^\beta$ .

So, under the (mild but unproven) heuristic that the blocks  $\mathbf{B}_{[1,j]}$  in a fully backward-reduced basis behave like a random matrices for all  $j$  (in terms of the number of redundant coordinates), it is easy to see that  $k_1 \gtrsim 2 \log_q n$ , which is significantly better than what LLL achieves (both in the worst case and empirically).

This heuristic argument is backed up by experiments. (See the full version [21].) We also show (in the full the version [21]) a less natural variant of this algorithm that provably achieves  $k_1 \gtrsim 2 \log_q n$  when its input is a random matrix. This variant works by carefully

<sup>6</sup> We argue (in the full version [21]) that there is not much point in taking  $\tau$  significantly greater than  $\log_q^2(n)$ .

<sup>7</sup> Computing a proper basis seems to require time  $O(nk^2 \log^2 q)$  (without using fast matrix multiplication algorithms), but in many contexts the input basis is in systematic form and is therefore proper.

“choosing which coordinates to look at” for each block, in order to maintain independence. We view this as an additional heuristic explanation for full backward reduction’s practical performance, since one expects an algorithm that “looks at all coordinates” to do better than one that does not.

This result about  $k_1$  for backward-reduced bases also compares favorably with the study of the LLL algorithm in [17]. In particular, in [17], they proved that LLL achieves  $k_1 \gtrsim \log n$  for a random code for  $q = 2$ , but in their experiments they observed that LLL combined with a preprocessing step called EpiSort actually seems to achieve  $k_1 \approx c \log n$  for some constant  $1 < c \leq 2$ . However, the behavior of LLL and EpiSort seems to be much more subtle than the behavior of full backwards reduction. We therefore still have no decent explanation (even a heuristic one) for why LLL and EpiSort seem to achieve  $k_1 \approx c \log n$  or for what the value of this constant  $c$  actually is.

#### 1.4.4 BKZ and slide reduction for codes

Our next set of contributions are adaptations of the celebrated BKZ and slide-reduction algorithms to the setting of codes.

##### 1.4.4.1 BKZ for codes

Our analogue of the BKZ algorithm for codes is quite natural.<sup>8</sup> Specifically, our algorithm works by repeatedly checking whether the epipodal vector  $\mathbf{b}_i^+$  is a shortest non-zero codeword in the code generated by the block  $\mathbf{B}_{[i, i+\beta-1]}$ . If not, it updates the basis so that this is the case (using the tools that we have developed to maintain properness). The algorithm does this repeatedly until no further updates are possible. At least intuitively, a larger choice of  $\beta$  here requires a slower algorithm because the resulting algorithm will have to find shortest non-zero codewords in  $\beta$ -dimensional codes. But, larger  $\beta$  will result in a better basis.

As we mentioned above, in the setting of lattices, the BKZ algorithm is considered to be the best performing basis-reduction algorithm in most parameter regimes, but it is notoriously difficult to analyze. We encounter a roughly similar phenomenon in the setting of linear codes. In particular, we run experiments that show that the algorithm performs quite well in practice. (Though it requires significantly more running time than full backward reduction to achieve a similar profile. See the full version [21].) However, we are unable to prove that it terminates efficiently, except in the special case of  $\beta = 2$ , in which case we recover the LLL algorithm of [17]. For  $\beta > 2$ , we offer only an extremely weak bound on the running time. As in the case of lattices, the fundamental issue is that it is difficult to control the effect that changing  $\mathbf{b}_i^+$  can have on the other epipodal vectors  $\mathbf{b}_{i+1}^+, \dots, \mathbf{b}_{i+\beta-1}^+$  in the block.

Here, we encounter an additional issue as well. In the case of lattices, there is a relatively simple tight bound on the minimum distance of the lattice generated by the block  $\mathbf{A}_{[i, i+\beta-1]}$  to the lengths of the Gram-Schmidt vectors  $\|\tilde{\mathbf{a}}_i\|, \dots, \|\tilde{\mathbf{a}}_{i+\beta-1}\|$  in the block. In particular, Minkowski’s celebrated theorem tells us that  $\lambda_1(\mathcal{L}(\mathbf{A}_{[i, i+\beta-1]})) \leq C\sqrt{\beta}(\|\tilde{\mathbf{a}}_i\| \cdots \|\tilde{\mathbf{a}}_{i+\beta-1}\|)^{1/\beta}$  for some constant  $C > 0$ , and one applies this inequality repeatedly with different  $i$  to understand the behavior of basis reduction for lattices.

<sup>8</sup> We note that the name “BKZ algorithm for codes” is perhaps a bit misleading. In the case of lattices, the BKZ algorithm is named after Korkine and Zolotarev due to their work on Korkine-Zolotarev-reduced bases (which can be thought of as BKZ-reduced bases with maximal block size  $\beta = k$ , and is sometimes also called a *Hermite*-Korkine-Zolotarev-reduced basis). A *Block*-Korkine-Zolotarev-reduced basis is (unsurprisingly) a basis in which each block  $\mathbf{B}_{[i, i+\beta-1]}$  is a Korkine-Zolotarev-reduced basis. For codes, the analogous notion of a Korkine-Zolotarev-reduced basis was called a Griesmer-reduced basis in [17]. So, we should perhaps call our notion “Block-Griesmer-reduced bases” and the associated algorithm “the block-Griesmer algorithm.” However, the authors decided to use the term “BKZ” here in an attempt to keep terminology more consistent between lattices and codes.

However, in the case of codes, there is no analogous simple tight bound on  $d_{\min}(\mathcal{C}(\mathbf{B}_{[i, i+\beta-1]}))$  in terms of the lengths of the epipodal vectors  $|\mathbf{b}_i^+|, \dots, |\mathbf{b}_{i+\beta-1}^+|$ , *except* in the special case when  $\beta = 2$ . Instead, there are many known incomparable upper bounds on  $d_{\min}$  in terms of the dimension  $\beta$  and the support size  $s := |\mathbf{b}_i^+| + \dots + |\mathbf{b}_{i+\beta-1}^+|$  (and, of course, the alphabet size  $q$ ). Each of these bounds is tight or nearly tight for some support sizes  $s$  (for fixed  $\beta$ ) but rather loose in other regimes. The nature of our basis-reduction algorithms is such that different blocks have very different support sizes  $s$ , so that we cannot use a single simple bound that will be useful in all regimes. And, due to the relatively “cramped” nature of  $\mathbb{F}_q^n$ , applying loose bounds on  $d_{\min}$  can easily yield trivial results, or results that do offer no improvement over the  $\beta = 2$  case. As a result, the bound that we obtain on the length of  $\mathbf{b}_1$  for a BKZ-reduced basis does not have a simple closed form. (Since the special case of  $\beta = 2$  yields a very simple tight bound  $d_{\min} \leq (1 - 1/q)s$ , this is not an issue in the analysis of the LLL algorithm in [17].)

In fact, we do not even know if the worst-case bound on  $|\mathbf{b}_1|$  for a BKZ-reduced basis is efficiently computable, even if one knows the optimal minimum distance of  $\beta$ -dimensional codes for all support sizes. However, we do show an efficiently computable bound that is nearly as good. And, we show empirically that in practice it produces quite a good basis. (See the full version [21].)

#### 1.4.4.2 Slide reduction for codes

Given our difficulties analyzing the BKZ algorithm, it is natural to try to adapt Gama and Nguyen’s slide-reduction algorithm [20] from lattices to codes. In particular, recall that in the case of lattices, the slide-reduction algorithm has the benefit that (unlike BKZ) it is relatively easy to prove that it terminates efficiently.

In fact, recall that the idea for backward-reduced bases was inspired by dual-reduced bases for lattices, which are a key component of slide reduction. We therefore define slide-reduced bases for codes by essentially just substituting backward-reduced blocks for dual-reduced blocks in Gama and Nguyen’s definition for lattices. Our slide-reduction algorithm (i.e., an algorithm that produces slide-reduced bases) follows similarly.

We then give a quite simple proof that this algorithm terminates efficiently. Indeed, our proof is a direct translation of Gama and Nguyen’s elegant potential-based argument from the case of lattices to the case of codes. (Gama and Nguyen’s proof is itself a clever variant of the beautiful original proof for the case when  $\beta = 2$  in [25].)

Finally, we give an efficiently computable upper bound on  $|\mathbf{b}_1|$  for a slide-reduced basis in a similar spirit to our upper bound on BKZ. Here, we again benefit from our analysis of backward-reduced blocks described above. Indeed, the behavior of the epipodal vectors in our backward-reduced blocks is quite easy to analyze. However, our bound does not have a simple closed form because the behavior of the forward-reduced blocks still depends on the subtle relationship between the minimal distance of a code and the parameters  $n$  and  $k$ , as we described in the context of BKZ above.

In our experiments (in the full version [21]), slide reduction is far faster than BKZ but does not find bases that are as good.

#### 1.4.5 Two illustrative algorithms

In the full version [21], we show yet two more basis-reduction algorithms for codes. We think of the importance of these algorithms as being less about their actual usefulness and more about what they show about the potential and limitations of basis reduction for codes. We explain below.

### 1.4.5.1 One-block reduction

The one-block-reduction algorithm is quite simple. It finds a short non-zero codeword in a code  $\mathcal{C}$  generated by some basis  $\mathbf{B}$  by first ensuring that  $\mathbf{B}$  is proper, and then by simply finding a shortest non-zero codeword in the subcode  $\mathcal{C}(\mathbf{B}_{[1,\beta]})$  generated by the prefix basis  $\mathbf{B}_{[1,\beta]}$ . Notice that if  $\beta \leq O(\log_q n)$ , then this algorithm runs in polynomial time. In particular, enumerating all codewords in the subcode can be done in time roughly  $O(nq^\beta \log q)$ .

Furthermore, it is not hard to see that when  $\beta = \lceil \log_q n \rceil$ , this simple algorithm actually meets the Griesmer bound! (See the full version [21].) At a high level, this is because (1) the worst case in the Griesmer bound has  $|\mathbf{b}_i^+| = 1$  for all  $i \geq \beta$ ; and (2) the resulting bound is certainly not better than the minimum distance of a code with dimension  $\beta$  and support size  $n - (k - \beta)$ . Here, the  $k - \beta$  term comes from the fact that  $\text{Supp}(\mathbf{B}_{[1,\beta]}) = n - |\mathbf{b}_{\beta+1}^+| - \dots - |\mathbf{b}_k^+|$ . (Similar logic explains why full backward reduction achieves the Griesmer bound with  $\tau \approx \log_q n$ .)

More generally, it seems unlikely that a basis-reduction algorithm will be able to find  $\mathbf{b}_1$  that is shorter than what is achieved by this simple approach if we take  $\beta \geq \max\{k_1^*, \beta'\}$ , where  $\beta'$  is the size of the largest block in the basis reduction algorithm and  $k_1^*$  is the maximal index of an epipodal vector that has length larger than one. (In practice,  $k_1^*$  is almost never much larger than  $k_1$ .) In particular, for a basis reduction algorithm to do better than this, it must manage to produce a block  $\mathbf{B}_{[1,\beta]}$  that has minimum distance less than what one would expect given its support size.

We therefore think of this algorithm as illustrating two points.

First, the existence of this algorithm further emphasizes the importance of the parameter  $k_1^*$  (and the closely related parameter  $k_1$ ) as a sort of “measure of non-triviality.” If an algorithm achieves large  $k_1^*$ , then the above argument becomes weaker, since we must take  $\beta \geq k_1^*$ . Indeed, if  $\beta$  is significantly larger than  $2 \log_q k$ , then the running time of one-block reduction (if implemented by simple enumeration) becomes significantly slower.

Second, the existence of the one-block-reduction algorithm illustrates that we should be careful not to judge basis-reduction algorithms *entirely* based on  $|\mathbf{b}_1|$ . We certainly think that  $|\mathbf{b}_1|$  is an important measure of study, and indeed it is the main way that we analyze the quality of our bases in this work. However, the fact that one-block-reduction exists shows that this should not be viewed as the only purpose of a basis-reduction algorithm.

Of course, the algorithms that we have discussed thus far are in fact non-trivial, because they (1) find short non-zero codewords *faster* than one-block reduction; and (2) find whole reduced bases and not just a single short non-zero codeword. Such reduced bases have already found exciting applications in [17] and [8], and we expect them to find more.

### 1.4.5.2 Approximate Griesmer reduction

Recall that [17] calls a basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  *Griesmer reduced* if  $\mathbf{b}_i^+$  is a shortest non-zero codeword in  $\mathcal{C}(\mathbf{B}_{[i,k]})$  for all  $i$ . And, notice that, if one is willing to spend the time to find shortest non-zero codewords in codes with dimension at most  $k$ , then one can compute a Griesmer-reduced basis iteratively, by first setting  $\mathbf{b}_1$  to be a shortest non-zero codeword in the whole code, then projecting orthogonal to  $\mathbf{b}_1$  and building the rest of the basis recursively. (Griesmer-reduced bases are the analogue of Korkine-Zolotarev bases for lattices. We discuss Griesmer-reduced bases more below.)

Our approximate-Griesmer-reduction algorithm is a simple variant of this idea. In particular, it is really a family of algorithms parameterized by a subprocedure that finds short (but not necessarily shortest) non-zero codewords in a code. Given such a subprocedure, the

algorithm first finds a short non-zero codeword  $\mathbf{b}_1$  in the input code  $\mathcal{C}$ . It then projects the code orthogonally to  $\mathbf{b}_1$  and builds the rest of the basis recursively. (To make sure that we end up with a proper basis, care must be taken to assure that  $\mathbf{b}_1$  is primitive. We ignore this in the introduction. See the full version [21].)

The running time of this algorithm and the quality of the basis produced of course depends on the choice of subprocedure. Given the large number of algorithms for finding short non-zero codewords with a large variety of performance guarantees for different parameters (some heuristic and some proven), we do not attempt here to study this algorithm in full generality. We instead simply instantiate it with the Lee-Brickell-Babai algorithm from [17] (an algorithm which itself uses [17]’s LLL algorithm as a subroutine). Perhaps unsurprisingly, we find that this produces significantly better basis profiles (e.g., smaller  $|\mathbf{b}_1|$  and larger  $k_1$  and  $k_1^*$ ) than all of the algorithms that we designed here. The price for this is, of course, that the subprocedure itself must run in enough time to find non-zero short codewords in dimension  $k$  codes.

We view this algorithm as a proof of concept, showing that at least in principle one can combine basis-reduction techniques with other algorithms for finding short codewords to obtain bases with very good parameters. This meshes naturally with the Lee-Brickell-Babai algorithm in [17], which shows how good bases can be combined with other algorithmic techniques to find short non-zero codewords. Perhaps one can merge these techniques more in order to show a way to use a good basis to find a better basis, which itself can be used to find a better basis, etc?

#### 1.4.6 On “the best possible bases”

Finally, in the full version [21], we prove bounds on “the best possible bases” in terms of the parameters  $k_1$  and  $k_1^*$ . Indeed, recall that the (heuristic) running time of [17]’s Lee-Brickell-Babai algorithm beats Lee-Brickell by a factor that is exponential in  $k_1$ . And, we argued above that  $k_1^*$  can be viewed as a measure of the “non-triviality” of a basis reduction algorithm. So, it is natural to ask how large  $k_1$  and  $k_1^*$  can be in principle.

In the full version [21], we show that *any* code over  $\mathbb{F}_2$  has a basis with  $k_1^* \geq \Omega(\log k^2)$ , even if the support size is as small as  $n = k + \sqrt{k}$ . For this, we use Griesmer-reduced bases (not to be confused with the approximate-Griesmer-reduced bases described above; note in particular that it is NP-hard to compute a Griesmer-reduced basis). Notice that this is a factor of  $\Omega(\log k)$  better than the logarithmic  $k_1^*$  achieved by all known efficient basis-reduction algorithms.

Here, we use the parameter  $k_1^*$  and not  $k_1$  because it is easy to see that in the worst case a code can have arbitrarily large support but still have no proper basis with  $k_1 > 1$ .<sup>9</sup> Typically, of course, one expects  $k_1^*$  and  $k_1$  to be very closely related, so that one can view this as heuristic evidence that typical codes have bases with  $k_1 \geq \Omega(\log^2 k)$ .

In the full version [21], we argue under a mild heuristic assumption that any basis for a random code over  $\mathbb{F}_2$  has  $k_1 \leq k_1^* \leq O(\log^2 k)$ , even if the support size  $n$  is a large polynomial in the dimension  $k$ .

Taken together, these results suggest that the best possible bases that we should expect to find in practice should have  $k_1 \approx k_1^* = \Theta(\log^2 k)$  for typical settings of parameters. Such a basis would (heuristically) yield a savings of  $k^{\Theta(\log k)}$  in [17]’s Lee-Brickell-Babai algorithm. So, it would be very exciting to find an efficient algorithm that found such a basis.

<sup>9</sup> For example, take that code  $\mathbb{F}_2^{k-1} \cup (\mathbb{F}_2^{k-1} + \mathbf{c})$  where  $\mathbf{c} := (1, 1, \dots, 1) \in \mathbb{F}_2^n$ . Any proper basis of this code must have  $k - 1$  vectors with length one and therefore must have  $k_1 = 1$ .



On the other hand, our (heuristic) upper bound on  $k_1$  suggests a limitation of basis reduction for codes. In particular, we should not expect any improvement better than  $k^{\Theta(\log k)}$  in Lee-Brickell-Babai. And, the upper bound also suggests that basis-reduction algorithms are unlikely to outperform the simple one-block-reduction algorithm for block sizes larger than  $\Omega(\log^2 k)$ .

## 2 Preliminaries

### 2.1 Some notation

Logarithms are base two unless otherwise specified, i.e.,  $\log(2^x) = x$ . We write  $\mathbf{I}_m$  for the  $m \times m$  identity matrix.

If  $\mathbf{b}_1, \dots, \mathbf{b}_k \in \mathbb{F}_q^n$ , then  $(\mathbf{b}_1, \dots, \mathbf{b}_k) \in \mathbb{F}_q^{n \times k}$  denotes the matrix where each  $\mathbf{b}_i$  is a column and  $(\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  denotes the matrix where each  $\mathbf{b}_i$  is row  $i$  of  $\mathbf{B}$ .

We say that a matrix  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  is *in systematic form* if  $\mathbf{A} = (\mathbf{I}_k, \mathbf{X})\mathbf{P}$ , where  $\mathbf{P}$  is a permutation matrix (i.e., if  $k$  contains the columns  $\mathbf{e}_1^T, \dots, \mathbf{e}_k^T$ ).

For any basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  and any subset  $S \subseteq [n]$  with  $|S| = k$  such that  $\mathbf{B}_S$  has full rank, we call the process of replacing  $\mathbf{B}$  by  $(\mathbf{B}|_S)^{-1}$  *systematizing  $\mathbf{B}$  with respect to  $S$* . When the set  $S$  is not important, we simply call this *systematizing  $\mathbf{B}$* . This procedure is useful at least in part because it results in a proper basis.

We define two notions of the support of a vector. Specifically, we write

$$\text{Supp}(\mathbf{x}) := \{i \in [1, n] : x_i \neq 0\},$$

and similarly

$$\overrightarrow{\text{Supp}}(\mathbf{x})_i := \begin{cases} 0 & x_i = 0 \\ 1 & x_i \neq 0. \end{cases}$$

We can also define the support of an  $[n, k]_q$  code  $\mathcal{C}$  by extending the definitions of  $\text{Supp}$  and  $\overrightarrow{\text{Supp}}$ ,

$$\text{Supp}(\mathcal{C}) \triangleq \bigcup_{c \in \mathcal{C}} \text{Supp}(c) \quad \overrightarrow{\text{Supp}}(\mathcal{C}) = \bigvee_{c \in \mathcal{C}} \overrightarrow{\text{Supp}}(c),$$

and we define the support of a matrix  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  as the support of the code generated by the matrix.

If  $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ ,  $\mathbf{B} \in \mathbb{F}_q^{r \times s}$ , then the direct sum of  $\mathbf{A}$  and  $\mathbf{B}$ , denoted  $\mathbf{A} \oplus \mathbf{B} \in \mathbb{F}_q^{(m+r) \times (n+s)}$ , is

$$\mathbf{A} \oplus \mathbf{B} = \begin{pmatrix} \mathbf{A} & \mathbf{0}_{m \times s} \\ \mathbf{0}_{n \times r} & \mathbf{B} \end{pmatrix}$$

We will often use the following important property regarding matrix direct sums. If  $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ ,  $\mathbf{B} \in \mathbb{F}_q^{r \times s}$ ,  $\mathbf{x} \in \mathbb{F}_q^n$ ,  $\mathbf{y} \in \mathbb{F}_q^s$ , then

$$(\mathbf{A} \oplus \mathbf{B}) \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{A}\mathbf{x} \\ \mathbf{B}\mathbf{y} \end{pmatrix}.$$

### 3 Generalizing epipodal vectors, size reduction, and the fundamental domain to $\mathbb{F}_q$

In this section, we generalize many of the fundamental concepts in [17] from codes over  $\mathbb{F}_2$  to codes over  $\mathbb{F}_q$ . Specifically, we generalize the notions of projection, epipodal matrices, and the size-reduction algorithm. We then study the geometry of the fundamental domain that one obtains by running the size-reduction algorithm on a given input basis.

Much of this generalization is straightforward (once one knows the theory developed for  $\mathbb{F}_2$  in [17]). So, one might read much of this section as essentially an extension of the preliminaries. The most difficult part, in the full version [21], is the analysis of the fundamental domain (which is not used in the rest of the paper).

#### 3.1 Projection and epipodal vectors

The notions of projection and epipodal vectors extend naturally to  $\mathbb{F}_q$  from the notions outlined in [17]. However, to ensure that this work is as self-contained as possible, we will now explicitly outline how some of those notions extend to  $\mathbb{F}_q$ . Notice that these operations are roughly analogous to orthogonal projection maps over  $\mathbb{R}^n$ .

► **Definition 1.** If  $\mathbf{x}_1 = (x_{1,1}, \dots, x_{1,n}), \dots, \mathbf{x}_k = (x_{k,1}, \dots, x_{k,n}) \in \mathbb{F}_q^n$ , the function  $\pi_{\{\mathbf{x}_1, \dots, \mathbf{x}_k\}} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$  is defined as follows:

$$\pi_{\{\mathbf{x}_1, \dots, \mathbf{x}_k\}}(\mathbf{y})_i = \begin{cases} y_i & x_{1,i} \neq 0 \vee \dots \vee x_{k,i} \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

We call this “projection onto the support of  $\mathbf{x}_1, \dots, \mathbf{x}_k$ .”

► **Definition 2.** If  $\mathbf{x}_1 = (x_{1,1}, \dots, x_{1,n}), \dots, \mathbf{x}_k = (x_{k,1}, \dots, x_{k,n}) \in \mathbb{F}_q^n$ , the function  $\pi_{\{\mathbf{x}_1, \dots, \mathbf{x}_k\}}^\perp : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$  is defined as follows:

$$\pi_{\{\mathbf{x}_1, \dots, \mathbf{x}_k\}}^\perp(\mathbf{y})_i = \begin{cases} y_i & x_{1,i} = 0 \wedge \dots \wedge x_{k,i} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We call this “projection orthogonal to  $\mathbf{x}_1, \dots, \mathbf{x}_k$ .”

We will often simply write  $\pi_{\mathbf{x}}$  to denote  $\pi_{\{\mathbf{x}\}}$  and  $\pi_{\mathbf{x}}^\perp$  to denote  $\pi_{\{\mathbf{x}\}}^\perp$ .

We now define the epipodal matrix of a basis for a code, which is the analogue of the Gram–Schmidt matrix.

► **Definition 3.** Let  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  be a matrix with elements from  $\mathbb{F}_q$ . The  $i$ th projection associated to the matrix  $\mathbf{B}$  is defined as  $\pi_i := \pi_{\{\mathbf{b}_1, \dots, \mathbf{b}_{i-1}\}}^\perp$ , where  $\pi_1$  denotes the identity.

The  $i$ th epipodal vector is defined as  $\mathbf{b}_i^+ := \pi_i(\mathbf{b}_i)$ . The matrix  $\mathbf{B}^+ := (\mathbf{b}_1^+; \dots; \mathbf{b}_k^+) \in \mathbb{F}_q^{k \times n}$  is called the epipodal matrix of  $\mathbf{B}$ .

The following notation for a projected block will be helpful in defining our reduction algorithms. (The same notation is used in the lattice literature.)

► **Definition 4.** For a basis  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  and  $i, j \in [1, k]$  where  $i \leq j$ , we use the notation  $\mathbf{B}_{[i,j]}$  as shorthand for  $(\pi_i(\mathbf{b}_i); \dots; \pi_i(\mathbf{b}_j))$ . Furthermore, for  $i \in [1, k]$  and  $j > k$ , we define  $\mathbf{B}_{[i,j]} = \mathbf{B}_{[i,k]}$  for all  $j > k$ .

We will often write  $\ell_i$  to denote  $|\mathbf{b}_i^+|$  when the basis  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k)$  is clear from context.

### 3.1.1 Basic operations on blocks

See the full version [21].

## 3.2 Size reduction and its fundamental domain

See the full version [21].

## 4 Proper bases and primitivity

We will primarily be interested in bases that are *proper* in the sense that the epipodal vectors should all be non-zero.

► **Definition 5.** *A basis is said to be proper if all its epipodal vectors  $\mathbf{b}_i^+$  are non-zero.*

[17] observed that, given an arbitrary basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  for a code, we can efficiently compute a proper basis  $\mathbf{B}'$  for the same code by systematizing  $\mathbf{B}$ . In particular, let  $\mathbf{A} \in \mathbb{F}_q^{k \times k}$  be an invertible matrix formed from  $k$  columns of  $\mathbf{B}$  (which must exist because  $\mathbf{B}$  is a full-rank matrix). Then,  $\mathbf{B}' := \mathbf{A}^{-1}\mathbf{B}$  is a proper basis for the code generated by  $\mathbf{B}$ . In particular, every code has a proper basis. From this, we derive the following simple but useful fact.

See the full version [21].

## 5 Redundant sets of coordinates, the last epipodal vector, and backward reduction

We are now ready to develop the theory behind backward-reduced bases. A backward-reduced basis is one in which the last epipodal vector  $\mathbf{b}_k^+$  is as *long* as possible. In the context of lattices, such bases are called dual-reduced bases and the maximal length of the last Gram-Schmidt vector has a simple characterization in terms of  $\lambda_1$  of the dual lattice. For codes, the maximal length of the last epipodal vector behaves rather differently, as we will explain below. In particular, we will see how to find a backward-reduced basis quite efficiently. In contrast, finding a dual-reduced basis is equivalent to finding a shortest non-zero vector in a lattice and is therefore NP-hard.

On our way to defining backward reduction, we first define the notion of *redundant* coordinates. Notice that we only consider coordinates in the support of  $\mathcal{C}$ .

► **Definition 6.** *For a code  $\mathcal{C} \subseteq \mathbb{F}_q^n$ , we say that a set  $S \subseteq [n]$  of coordinates is redundant for  $\mathcal{C}$  if  $S \subseteq \text{Supp}(\mathcal{C})$  and for every  $\mathbf{c} \in \mathcal{C}$  and all  $i, j \in S$ ,  $c_i = 0$  if and only if  $c_j = 0$ .*

The following simple claim explains the name “redundant.” In particular, for any codeword  $\mathbf{c} \in \mathcal{C}$ , if we know  $c_i$  for some  $i \in S$ , then we also know  $c_j$  for any  $j \in S$ .

▷ **Claim 7.** For a code  $\mathcal{C} \subseteq \mathbb{F}_q^n$ , a set  $S \subseteq \text{Supp}(\mathcal{C})$  is redundant for  $\mathcal{C}$  if and only if for every  $i, j \in S$ , there exists a non-zero scalar  $a \in \mathbb{F}_q^*$  such that for all  $\mathbf{c} \in \mathcal{C}$ ,  $c_j = ac_i$ .

Furthermore, to determine whether  $S$  is a set of redundant coordinates, it suffices to check whether the latter property holds for all  $\mathbf{c} := \mathbf{b}_i$  in a basis  $(\mathbf{b}_1; \dots; \mathbf{b}_k)$  of  $\mathcal{C}$ .

Proof. See the full version [21].

◁

Next, we show that redundancy is closely connected with the last epipodal vector in a basis.

## 19:18 More Basis Reduction for Linear Codes

► **Lemma 8.** For a code  $\mathcal{C} \subseteq \mathbb{F}_q^n$  with dimension  $k$  and  $S \subseteq [n]$ , there exists a basis  $\mathbf{B} := (\mathbf{b}_1; \dots; \mathbf{b}_k)$  of  $\mathcal{C}$  with  $S \subseteq \text{Supp}(\mathbf{b}_k^+)$  if and only if  $S$  is redundant.

Furthermore, if  $S$  is redundant, then there exists a proper basis with this property.

**Proof.** See the full version [21]. ◀

The above motivates the following definition.

► **Definition 9.** For a code  $\mathcal{C} \subseteq \mathbb{F}_q^n$ , the repetition number of  $\mathcal{C}$ , written  $\eta(\mathcal{C})$ , is the maximal size of a redundant set  $S \subseteq \text{Supp}(\mathcal{C})$ .

In particular, notice that by Lemma 8,  $\eta(\mathcal{C})$  is also the maximum of  $|\mathbf{b}_k^+|$  over all bases  $(\mathbf{b}_1; \dots; \mathbf{b}_k)$  and this maximum is achieved by a proper basis. The next lemma gives a lower bound on  $\eta(\mathcal{C})$ , therefore showing that codes with sufficiently large support and sufficiently low rank must have bases whose last epipodal vector is long.

► **Lemma 10.** For any code  $\mathcal{C} \subseteq \mathbb{F}_q^n$  with dimension  $k$ ,

$$\eta(\mathcal{C}) \geq \left\lceil \frac{q-1}{q^k-1} \cdot |\text{Supp}(\mathcal{C})| \right\rceil.$$

**Proof.** See the full version [21]. ◀

We present in Algorithm 1 a simple algorithm that finds the largest redundant set of a code  $\mathcal{C}$ . (The algorithm itself can be viewed as a constructive version of the proof of Lemma 10.)

■ **Algorithm 1** Max Redundant Set.

---

**Input:** A basis  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  for  $\mathcal{C}$

**Output:** A redundant set  $S$  for  $\mathcal{C}$  with  $|S| = \eta(\mathcal{C})$

**for**  $j \in [n]$  **do**

  |  $a_j \leftarrow B_{i,j}^{-1}$ , where  $i \in [k]$  is minimal such that  $B_{i,j} \neq 0$ .

**end**

Find  $S \subseteq \text{Supp}(\mathcal{C})$  with maximal size such that for all  $j_1, j_2 \in S$  and all  $i$ ,

$a_{j_1} B_{i,j_1} = a_{j_2} B_{i,j_2}$ .

**return**  $S$

---

▷ **Claim 11.** Algorithm 1 outputs a redundant set  $S$  for  $\mathcal{C}$  with  $|S| = \eta(\mathcal{C})$ . Furthermore, Algorithm 1 runs in time  $O(kn \log(q) \log(qn))$  (when implemented appropriately).

**Proof.** See the full version [21]. ◀

### 5.1 Backward reduction

We are now ready to present our definition of backward-reduced bases.

► **Definition 12.** Let  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  be a basis of a code  $\mathcal{C}$ . We say that  $\mathbf{B}$  is backward reduced if it is proper and  $|\mathbf{b}_k^+| = \eta(\mathcal{C}(\mathbf{B}))$ .

Finally, we give an algorithm that finds a backward-reduced basis. See Algorithm 2.

▷ **Claim 13.** On input a proper basis  $\mathbf{B}$ , Algorithm 2 correctly outputs an invertible matrix  $\mathbf{A}$  such that  $\mathbf{A}\mathbf{B}$  is backward reduced. Furthermore, Algorithm 2 runs in time  $O(nk \log(q) \log(qn))$ .

**Proof.** See the full version [21]. ◀

■ **Algorithm 2** Backward Reduction.

---

**Input:** A proper basis  $\mathbf{B} = (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  for  $\mathcal{C}$   
**Output:** An invertible matrix  $\mathbf{A} \in \mathbb{F}_q^{k \times k}$  such that  $\mathbf{A}\mathbf{B}$  is backward reduced.  
 $\{j_1, \dots, j_t\} \leftarrow \text{MaxRedundantSet}(\mathbf{B})$   
Let  $m$  be minimal such that  $B_{m,j_1} \neq 0$ .  
**for**  $i \in [m+1, k]$  **do**  
  |  $\mathbf{b}_i \leftarrow \mathbf{b}_i - B_{m,j_1}^{-1} B_{i,j_1} \mathbf{b}_m$   
**end**  
 $(\mathbf{b}_1; \dots; \mathbf{b}_k) \leftarrow (\mathbf{b}_1; \dots; \mathbf{b}_{m-1}; \mathbf{b}_{m+1}; \dots; \mathbf{b}_k; \mathbf{b}_m)$   
**return** the matrix corresponding to the linear transformation done to  $\mathbf{B}$ .

---

## 5.2 Full backward reduction

Since backward reduction can be done efficiently, it is natural to ask what happens when we backward reduce many prefixes  $\mathbf{B}_{[1,i]}$  of a basis. We could simply do this for all  $i \in [k]$ , but it is natural to be slightly more fine-grained and instead only do this for  $i \leq \tau$  for some threshold  $\tau$ . In particular, since the last  $k - \text{poly}(\log n)$  epipodal vectors tend to have length one even in very good bases (see the full version [21] to understand why), it is natural to take  $\tau \leq \text{poly}(\log n)$  to be quite small, which leads to very efficient algorithms. This suggests the following definition.

► **Definition 14.** For some threshold  $\tau \leq k$ , a basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  is fully backward reduced up to  $\tau$  if it is proper and  $\mathbf{B}_{[1,i]}$  is backward reduced for all  $1 \leq i \leq \tau$ .

We now show how to easily and efficiently compute a fully backward-reduced basis, using the backward-reduction algorithm (Algorithm 2) that we developed above. We present the algorithm in Algorithm 3 and then prove its correctness and efficiency. Notice in particular that the algorithm only changes each prefix  $\mathbf{B}_{[1,i]}$  (at most) once.

■ **Algorithm 3** Full Backward Reduction.

---

**Input:** A proper basis  $\mathbf{B} := (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  for a code  $\mathcal{C}$  and a threshold  $\tau \in [1, k]$   
**Output:** A basis for  $\mathcal{C}$  that is totally backward reduced up to  $\tau$ .  
**for**  $i = \tau, \dots, 1$  **do**  
  |  $\mathbf{A} \leftarrow \text{BackwardReduction}(\mathbf{B}_{[1,i]})$   
  |  $\mathbf{B} \leftarrow (\mathbf{A} \oplus \mathbf{I}_{k-i})\mathbf{B}$   
**end**  
**return**  $\mathbf{B}$

---

► **Theorem 15.** On input a proper basis  $\mathbf{B} := (\mathbf{b}_1; \dots; \mathbf{b}_k) \in \mathbb{F}_q^{k \times n}$  for a code  $\mathcal{C}$  and a threshold  $\tau \in [1, k]$ , Algorithm 3 correctly outputs a basis  $\mathbf{B}' \in \mathbb{F}_q^{k \times n}$  for  $\mathcal{C}$  that is fully backward reduced up to  $\tau$ . Furthermore, the algorithm runs in time  $O(\tau^2 n \log(q) \log(qn))$ .

**Proof.** See the full version [21]. ◀

We next bound  $|\mathbf{b}_1|$  of a fully backward-reduced basis. In fact, when  $\tau \geq \lceil \log_q n \rceil$ , this bound matches the Griesmer bound. In fact, it is not hard to see that with  $\tau = k$ , a fully backward-reduced basis is in fact LLL-reduced as well. But, the below theorem shows that we do not need to go all the way to  $\tau = k$  to achieve the Griesmer bound. This is because in the worst case,  $|\mathbf{b}_i^+| = 1$  for all  $i \geq \log_q n$  anyway.

► **Theorem 16.** *For any positive integers  $k$ ,  $n \geq k$ , and  $\tau \leq k$ , a basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$  of a code  $\mathcal{C}$  that is fully backward reduced up to  $\tau$  satisfies*

$$\sum_{i=1}^{\tau} \left\lceil \frac{|\mathbf{b}_i|}{q^{i-1}} \right\rceil \leq n - k + \tau .$$

**Proof.** See the full version [21]. ◀

### 5.3 Heuristic analysis suggesting better performance in practice

Recall that our analysis of backward-reduced bases in Section 5 relied crucially on the repetition number  $\eta(\mathcal{C})$ , which is the maximum over all bases of  $\mathcal{C}$  of the last epipodal vector. We showed that  $\eta(\mathcal{C})$  can be equivalently thought of as the maximal set of redundant coordinates. E.g., when  $q = 2$ ,  $\eta(\mathcal{C})$  is precisely the number of repeated columns in the basis  $\mathbf{B}$  for  $\mathcal{C}$ .

Our analysis of fully backward-reduced bases then relies on the lower bound on  $\eta(\mathcal{C})$  in Lemma 10. The proof of Lemma 10 simply applies the pigeonhole principle to the (normalized, non-zero) columns of a basis  $\mathbf{B}$  for  $\mathcal{C}$  to argue that, if there are enough columns, then one of them must be repeated many times. Of course, the pigeonhole principle is tight in general and it is therefore easy to see that this argument is tight in the worst case.

However, in the average case, this argument is not tight. For example, if the number  $n$  of (non-zero) columns in our basis  $\mathbf{B} \in \mathbb{F}_2^{k \times n}$  is smaller than the number of possible (non-zero) columns  $2^k - 1$ , then it is certainly possible that no two columns will be identical. But, the birthday paradox tells us that even with just  $n \approx 2^{k/2}$ , a random matrix  $\mathbf{B} \in \mathbb{F}_2^{k \times n}$  will typically have a repeated column. More generally, if a code  $\mathcal{C}$  is generated by a random basis  $\mathbf{B} \in \mathbb{F}_q^{k \times n}$ , then we expect to have  $\eta(\mathcal{C}) > 1$  with probability at least  $1 - 1/\text{poly}(n)$ , provided that, say,  $n \geq 10 \log(n)q^{k/2}$ , or equivalently, provided that

$$k \leq 2(\log_q n - \log_q(10 \log(n))) .$$

We could now make a heuristic assumption that amounts to saying that the prefixes  $\mathbf{B}_{[1,i]}$  behave like random matrices with suitable parameters (in terms of the presence of repeated non-zero columns). We could then use such a heuristic to show that we expect the output of Algorithm 3 to achieve

$$k_1 > (2 - o(1)) \log_q n .$$

We choose instead to present in the full version [21] a variant of Algorithm 3 that provably achieves the above. This variant is identical to Algorithm 3 except that instead of looking at all of  $\mathbf{B}_{[1,i]}$  and choosing the largest set of redundant coordinates in order to properly backward reduce  $\mathbf{B}_{[1,i]}$ , the modified algorithm chooses the largest set of redundant coordinates *from some small subset* of all of the coordinates. In other words, the modified algorithm ignores information. Because the algorithm ignores this information, we are able to rigorously prove that the algorithm achieves  $k_1 \gtrsim 2 \log_q n$  when its input is a random matrix (by arguing that at each step the algorithm has sufficiently many fresh independent random coordinates to work with).

We think it is quite likely that Algorithm 3 performs better (and certainly not much worse) than this information-ignoring variant. We therefore view this as strong heuristic evidence that Algorithm 3 itself achieves  $k_1 \gtrsim 2 \log_q n$ . (This heuristic is also confirmed by experiment. See the full version [21].)

### 5.3.1 Backward reducing without all of the columns

See the full version [21].

---

#### References

- 1 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in  $2^n$  time using discrete Gaussian sampling. In *STOC*, 2015.
- 2 Divesh Aggarwal and Noah Stephens-Davidowitz. (Gap/S)ETH hardness of SVP. In *STOC*, 2018.
- 3 Miklós Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.
- 4 Miklós Ajtai. The Shortest Vector Problem in L2 is NP-hard for randomized reductions. In *STOC*, 1998.
- 5 Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, 1997.
- 6 Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the Shortest Lattice Vector Problem. In *STOC*, pages 601–610, 2001.
- 7 Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, pages 298–307, 2003.
- 8 Nicolas Aragon, Julien Lavauzelle, and Matthieu Lequesne. [decodingchallenge.org](http://decodingchallenge.org), 2019. URL: <http://decodingchallenge.org>.
- 9 Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.
- 10 L. Babai. On Lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
- 11 Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In *Asiacrypt*, 2018.
- 12 Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *SODA*, 2016.
- 13 Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. On the quantitative hardness of CVP. In *FOCS*, 2017.
- 14 E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.
- 15 Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical report, University of Amsterdam, 1981.
- 16 Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *Asiacrypt*, 2011.
- 17 Thomas Debris-Alazard, Léo Ducas, and Wessel P. J. van Woerden. An algorithmic reduction theory for binary codes: LLL and more. *IEEE Transactions on Information Theory*, 68(5):3426–3444, 2022. URL: <https://eprint.iacr.org/2020/869>.
- 18 Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003.
- 19 I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.
- 20 Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell’s inequality. In *STOC*, 2008.
- 21 Surendra Ghentiyala and Noah Stephens-Davidowitz. More basis reduction for linear codes: backward reduction, BKZ, slide reduction, and more, 2024.
- 22 J. H. Griesmer. A bound for error-correcting codes. *IBM Journal of Research and Development*, 4(5):532–542, 1960.
- 23 Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

## 19:22 More Basis Reduction for Linear Codes

- 24 P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Eurocrypt*, 1988.
- 25 Arjen K. Lenstra, Hendrik W. Lenstra, Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, December 1982.
- 26 Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Propulsion Laboratory, 1978.
- 27 Daniele Micciancio. The Shortest Vector Problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2001.
- 28 Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In *Eurocrypt*, 2016. URL: <http://eprint.iacr.org/2015/1123>.
- 29 Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm: Survey and Applications*. Springer-Verlag, 2010.
- 30 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):Art. 34, 40, 2009. doi:10.1145/1568318.1568324.
- 31 Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(23):201–224, 1987.
- 32 Noah Stephens-Davidowitz and Vinod Vaikuntanathan. SETH-hardness of coding problems. In *FOCS*, 2019.
- 33 Alexander Vardy. Algorithmic complexity in coding theory and the Minimum Distance Problem. In *STOC*, 1997.
- 34 Michael Walter. Lattice blog reduction: The Simons Institute blog. <https://blog.simons.berkeley.edu/2020/04/lattice-blog-reduction-part-i-bkz/>, 2020.



# Online $k$ -Median with Consistent Clusters

Benjamin Moseley  

Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA

Heather Newman  

Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA

Kirk Pruhs  

Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

---

## Abstract

We consider the problem in which  $n$  points arrive online over time, and upon arrival must be irrevocably assigned to one of  $k$  clusters where the objective is the standard  $k$ -median objective. Lower-bound instances show that for this problem no online algorithm can achieve a competitive ratio bounded by *any* function of  $n$ . Thus we turn to a beyond worst-case analysis approach, namely we assume that the online algorithm is a priori provided with a predicted budget  $B$  that is an upper bound to the optimal objective value (e.g., obtained from past instances). Our main result is an online algorithm whose competitive ratio (measured against  $B$ ) is solely a function of  $k$ . We also give a lower bound showing that the competitive ratio of every algorithm must depend on  $k$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Online algorithms

**Keywords and phrases**  $k$ -median, online algorithms, learning-augmented algorithms, beyond worst-case analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.20

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2303.15379> [20]

**Funding** *Benjamin Moseley:* Supported in part by a Google Research Award, an Inform Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146.

*Heather Newman:* Supported in part by a Google Research Award, an Inform Research Award, a Carnegie Bosch Junior Faculty Chair, and NSF grants CCF-2121744 and CCF-1845146.

*Kirk Pruhs:* Supported by NSF grants CCF-1907673, CCF-2036077, CCF-2209654 and an IBM Faculty Award.

## 1 Introduction

Clustering problems, such as  $k$ -means clustering and  $k$ -median clustering, are a classic genre of learning / data mining problems [5]. Typically the input consists of a collection  $X = \{x_1, \dots, x_n\}$  of points in some metric space  $\mathcal{M}$  (typically  $\mathbb{R}^d$  with the 1-norm or 2-norm) and a positive integer  $k$ . Typically  $k$  is a small constant [2, 5]. The output for a **center-based clustering problem** is a collection  $c_1, \dots, c_k$  of  $k$  points from  $X$ , called centers, that succinctly summarize the data points. The implicit cluster  $C_i$  corresponding to the center  $c_i$  is the collection of points in  $X$  whose closest center is  $c_i$ , that is  $C_i = \{x_j \mid \arg \min_{h \in [k]} d(x_j, c_h) = i\}$ , where  $d(\cdot, \cdot)$  is the distance function for the metric space. The output for a **cluster-based clustering problem** is a partition  $C_1, \dots, C_k$  of  $X$  into  $k$  parts, called clusters. The implicit center of each cluster  $C_i$  is then  $c_i = \arg \min_{x_h \in C_i} \sum_{x_j \in C_i} d(x_h, x_j)$ . For both center-based clustering and cluster-based clustering, the objective is to minimize the cost of the clustering. This paper considers the  $k$ -median objective which is the aggregate distance from each point to the center of its cluster, that is  $\sum_{i=1}^k \sum_{x_j \in C_i} d(x_j, c_i)$ .



© Benjamin Moseley, Heather Newman, and Kirk Pruhs;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 20; pp. 20:1–20:22



Leibniz International Proceedings in Informatics

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

Here we consider applications where the data points in  $X$  arrive online over time. In an **online center-based clustering problem**, the online algorithm maintains a collection of centers. In a **online cluster-based clustering problem**, the online algorithm needs to assign the data points to a cluster when they arrive, that is each point  $x_j$  needs to be assigned a label  $\ell_j \in [k]$  when  $x_j$  arrives. In either case, these choices should (ideally) be irrevocable.

An application of online clustering given by [16] is the task of clustering news articles that arrive online, e.g., at Yahoo news or Google news. We refer to these outlets as the news providers. The news provider selects some (approximately) fixed number  $k$  of articles to feature on the news homepage, and has a “view complete coverage” link next to each article to see all the news stories on this topic. The problem of selecting the best  $k$  articles that summarize all current news articles is better modeled as a center-based clustering. The problem of partitioning all news articles into clusters of similar articles is better modeled as a cluster-based clustering. Other applications can be found in [17, 14], but we will use the news story clustering application as our running canonical example.

A line of research [12, 11, 14, 17] within online clustering goes by moniker of consistent clustering. Research on consistent clustering studies the tradeoffs between the following objectives:

- **Maximizing the Quality of the Clustering:** One seeks a clustering of small cost. The most common metric to measure the quality of a solution is the ratio of the cost of this solution to cost of the optimal solution. The most common metric to measure the quality of an online algorithm is the competitive ratio, which is the maximum (over all inputs) of the ratio of the cost of the online algorithm’s solution to the optimal cost.
- **Maximizing Consistency:** Ideally one would like the centers in a center-based problem, or the clusters (labels of points) in a cluster-based problem, to be consistent over time. That is, they should change as little as possible. E.g., the news provider does not want the clusters to completely change every time a new news article is written.

## 1.1 Prior Work on Consistent Clustering

$k$ -median clustering is NP-hard, but constant-factor approximation algorithms are known [9, 13, 1, 15, 6].

All prior algorithmic research on consistent clustering that we are aware of [12, 11, 14, 17, 4] is center-based. That is, the online algorithm explicitly maintains a collection of centers, and the clustering is implicit; each point is associated with the closest center, but there are no restrictions on how often points’ associated centers can change.

In the first paper in this line of research, Liberty et al. [17] gave a lower bound that showed that one cannot simultaneously have both high quality and maximum consistency. That is, they showed that if a center cannot be changed once it is established, then there is no algorithm whose competitive ratio can be bounded by any function of  $n$  and  $k$ . Thus various “beyond worst-case analysis” (see [21]) approaches have been used in the literature to attempt to circumvent the obstacle presented by this lower bound. One approach is to use bi-criteria analysis or *resource augmentation* analysis. This analysis allows the online algorithm to use more than  $k$  centers, and then compares the cost of the algorithm’s clustering to the optimal one using  $k$  centers [17]. A second approach is to allow the algorithm *recourse*, which in this setting means allowing the algorithm to change the centers (or clusters) a small number of times [14, 11, 12].

**Resource Augmentation.** Liberty et al. [17] give a randomized algorithm for  $k$ -means clustering and analyzes this algorithm using *resource augmentation* analysis. They show that the expected number of clusters/centers used by their algorithm is  $O(k \log n \log(n\Delta))$  and at all times the expected cost of the clustering using these centers is at most  $O(\log n)$  times the optimal cost using  $k$  clusters. Here  $\Delta$  is the aspect ratio of the data points, which is the ratio between the distance between the furthest pair of points and the distance between the closest pair of points. The algorithm leverages a randomized online algorithm for facility location of Meyerson [18] to decide whether to create a new center at a newly arriving data point. Once a center is established, it is maintained throughout the course of the algorithm. Finally, they give a randomized algorithm that requires a priori knowledge of  $n$  and a lower bound on the optimal with  $k$  centers, and that maintains a collection of  $O(k \log n \log \alpha)$  centers in expectation that has expected cost  $O(1)$  times the optimal cost with  $k$  centers. Here  $\alpha$  is the ratio between the actual optimal cost with  $k$  centers and the lower bound provided a priori to the algorithm.

**Recourse.** Lattanzi and Vassilvitskii [14] give a randomized algorithm for  $k$ -median clustering that uses *recourse*. It maintains the invariant that the cost of the current centers is always  $O(1)$ -competitive with the optimal clustering of the data points seen *to date*. To maintain this invariant, the expected number of cluster center changes used is  $O(k^2 \log^4(n\Delta))$ . They show a similar lower bound, that is they show that every algorithm requires  $\Omega(k \log_c \frac{\Delta}{k})$  center changes to maintain  $O(c)$ -competitiveness. Further, they show that it possible to maintain  $O(1)$ -competitiveness with  $O(k \log^2(n\Delta))$  center changes, but this is given as an existential result. In a follow-up paper, Fichtenberger et al. [11] gave a randomized algorithm that is  $O(1)$ -competitive with  $O(k \text{polylog}(n\Delta))$  cluster center changes. In both papers, the result of Meyerson [18] is again a key subroutine. The results of Lattanzi and Vassilvitskii [14] were extended to  $k$ -median clustering with outliers (so one could opt to not cluster a pre-specified number of points) by Guo et al. [12]. Lattanzi and Vassilvitskii [14] also observe that for  $k$ -center clustering an algorithm of Charikar et al. [8] yields an  $O(1)$ -competitive clustering with  $O(k \log(n\Delta))$  center changes.

While not directly germane to the work in this paper, there is also research on online clustering in the streaming setting, where the emphasis is more on the algorithm using a small amount of memory, or quickly responding to the arrival of a new data point (e.g. [7, 10, 3]).

## 1.2 Our Contribution

Our research investigates consistent clustering for cluster-based problems (recall that all the past algorithmic consistent clustering publications that we are aware of focus on center-based clustering). We are interested in applications where the focus is on explicitly maintaining consistent clusters (and not necessarily on maintaining consistent centers). The application where Google or Yahoo news is trying to maintain collections of similar news articles is an example of such an application. Note that even the algorithms from [17] that are perfectly consistent from a center perspective, in that once a center is established it persists until the end of the algorithm, are not necessarily consistent from a cluster perspective in that a data point could change clusters every time a new center is established. All one can say (at least naively) about the cluster consistency of the algorithms from [17] is that no data point changes clusters more than  $O(k \log n \log(n\Delta))$  times.

► **Problem 1** ((Online) cluster-based clustering). *The points  $X = \{x_1, \dots, x_n\}$  from a metric space arrive online in this (adversarial) order. Each point must be given an irrevocable label from  $\{1, \dots, k\}$  (i.e., irrevocably assigned a cluster) upon arrival. (The number of points  $n$  is not known.) The goal is to minimize the  $k$ -median objective.*

**Beyond Worst-Case Model.** We first observe that the lower bound from [17] extends to the cluster-based setting, so no online algorithm can achieve a competitive ratio bounded by *any* function of  $n$ . Thus we turn to a learning-augmented approach, namely we assume that the algorithm is provided a priori with an estimated upper bound  $B$  on the cost  $\text{OPT}$  of the final optimal clustering; recourse and resource augmentation are **not** allowed. This approach is both natural and appealing, as the a priori information provided to the algorithm is *minimal*. Moreover, it finds motivation in the Google/Yahoo news application, where presumably the final objective values for prior instances could be used as a basis to obtain a reasonable estimate for  $B$ . Thus we then seek algorithms that will maintain a clustering of low cost relative to  $B$  (not the current optimal cost for the points that have arrived to date). We say an algorithm is  $c$ -competitive with respect to  $B$  if the algorithm's cost is at most  $c \cdot B$  on instances where the optimal cost is at most  $B$  (after all points have arrived).

We first show that any deterministic algorithm must have dependence on  $k$  in the competitive ratio. The proof is deferred to the full version.

► **Proposition 1.** *Any deterministic algorithm for cluster-based clustering is  $\Omega(k)$ -competitive with respect to  $B$ .*

In almost all applications,  $k$  is a small constant [2, 5]. Thus, we ask if an algorithm can have performance only depending on  $k$  and not on the large parameters  $\Delta$  and  $n$ .

*Does there exist an online algorithm for Problem 1 that, given a priori knowledge of an upper bound  $B$  on  $\text{OPT}$ , achieves competitiveness independent of  $n$  and  $\Delta$ ?*

### 1.3 Results

Our main question is whether there exists an algorithm with competitiveness depending only on  $k$  (and not  $n$  or  $\Delta$ ). We answer this constructively:

► **Theorem 2.** *There is a poly-time algorithm for cluster-based clustering that is  $O(k^5 3^k)$ -competitive with respect to  $B$ .*

Intuitively, our algorithm uses the value of  $B$  to determine a scale for which costs are cheap (namely that are small relative to  $B$ ) and which are expensive (namely that are large relative to  $B$ ). Thus, one upshot of our results is that this minimal scaling information is all that the online algorithm needs to overcome the strong lower bound. Moreover, existing algorithms/subroutines in the recourse and resource augmentation settings do not seem to translate to guarantees in our setting. Thus, our setting requires novel techniques and structural insights, which we turn to next.

## 2 Technical Overview

As our algorithm is fairly detailed, we begin with a technical overview to build the case for our design decisions.

To understand the motivation for the learning-augmented approach, let us consider the lower bound instance from [17]. It is sufficient to assume  $k = 2$ . The first point  $x_1$  arrives and is assigned some irrevocable label. Then assume the second data point  $x_2$  arrives a unit

distance from  $x_1$ . If the online algorithm assigns  $x_2$  the same label as  $x_1$ , then the cost of the algorithm's clustering is 1, and the optimal cost is 0 (which is the cost if each of these data points were given a different label). This results in the algorithm having unbounded competitiveness. In contrast, if the algorithm gave  $x_2$  a different label from  $x_1$  then the third data point  $x_3$  could arrive very far away. In which case, the algorithm's clustering would necessarily have very high cost (as  $x_3$ 's label would have to be either the same as  $x_1$ 's or the same as  $x_2$ 's). However, the optimal clustering would have cost 1 (by giving  $x_1$  and  $x_2$  the same label and giving  $x_3$  the remaining label). Again, this results in competitiveness that can only be bounded by  $\Delta$  (which may be much larger than  $n$  or  $k$ ).

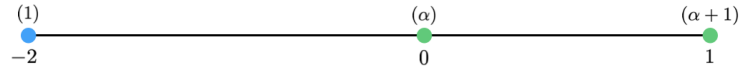
Intuitively, the dilemma faced by the algorithm when  $x_2$  arrives is that it does not know whether the distance between  $x_1$  and  $x_2$  is small or large. Equipped with an estimate of the optimal cost  $B$ , the algorithm could resolve the dilemma in this case by giving  $x_2$  a different label than  $x_1$  if their distance is larger than  $B$  and the same label otherwise.

## 2.1 Properties of competitive algorithms

To better understand our algorithm design it is useful to understand some instances that illustrate some properties that a competitive algorithm must have.

A simple first observation is that any reasonably competitive algorithm can never use  $t + 1$  labels if it is the case that the points to date could be clustered with cost at most  $B$  using at most  $t$  labels. If the algorithm ever allowed this to happen, it could be that the next  $k - t$  data points could arrive very far from the previous data points, and very far from each other. Thus after these data points arrive, the algorithm's cost would increase by an amount depending on the diameter of the metric space, while there would still be a clustering of cost at most  $B$ , since the clustering that used  $t$  labels could be extended with no additional cost by giving each of the new  $k - t$  data points a new label.

**Natural greedy algorithm fails.** In light of this last observation, a natural greedy algorithm would maintain the invariant that the number of labels it uses is always equal to the minimum number of labels necessary for a clustering of cost at most  $B$ , and then give each new data point the label that minimizes the increase in cost. To see why such an algorithm (and other similar algorithms) can have unbounded cost even when  $k = 2$ , and the metric space is the real line, consider the following instance (see Figure 1). Let  $\alpha$  be an arbitrarily large positive integer. We construct an example in which the budget  $B = 2$ . The first point arrives at location  $-2$ . Say the algorithm gives this point the label blue. The next point arrives at location 1. Now, we know that any offline clustering with cost at most 2 must use at least 2 clusters. So the greedy algorithm would give this new point a second label, say green, as that would minimize the increase in the objective. Then a total of  $\alpha$  additional points arrive at location 1. The algorithm labels these points green. Then  $\alpha$  points arrive at the origin 0. It is still the case that only 2 clusters are required in order to have cost at most 2, since we may place the points at location  $-2$  and the origin in one cluster, and the points at location 1 in the other cluster. However the algorithm would assign each point arriving at the origin the label green, since this increases the objective by at most 1 while assigning such a point the label blue increases the objective by 2. Yet, this results in a solution for the algorithm in which the contribution of green points towards the objective is  $\alpha$ .



■ **Figure 1** An example in which the natural greedy algorithm fails.

Upon reflection of this lower bound instance for the natural greedy algorithm, there appear to us to be two natural hypotheses as to the “mistake” that this algorithm is making, and correspondingly two natural paths towards a remedy:

- One hypothesis is that greedy assignment is a mistake, and then the natural remedy is some label assignment rule more sophisticated than greedy.
- Another is that the algorithm was too hasty in using a new label. Thus the natural remedy would be to delay using a new label until it is more clear as to a region where arriving data points should be given this new label. Note in the example in Figure 1 that if the algorithm had waited until some reasonable number of data points had arrived at the origin before using the second label, then the algorithm might have been able to see that the right choice was to give the remaining points arriving at the origin the second label of green.

## 2.2 Techniques

Here we primarily adopt the second remedy/approach (while also considering an alternate greedy assignment policy). To apply this remedy we must address the following:

- Under what conditions can the algorithm justify the use of an additional label, increasing from  $t - 1$  labels to  $t$  labels?
- When this can be justified, how should we modify our prior partition of space into  $t - 1$  parts to a partition into  $t$  parts? We would like to greedily assign each point to its “closest” part.

**Well-separated points.** At a high level our answer to the first question is that we do not use  $t$  labels until there exist  $t$  well-separated points  $x_{\alpha(1)}, \dots, x_{\alpha(t)}$ . We will say that a collection of points  $x_{\alpha(1)}, \dots, x_{\alpha(t)}$  from a collection  $S$  of points is  **$\beta$ -well-separated with respect to  $w_S$**  (for some  $\beta > 0$ ) if for all  $i, j \in [t], i \neq j$

$$\min\{w_S(x_{\alpha(i)}), w_S(x_{\alpha(j)})\} \cdot d(x_{\alpha(i)}, x_{\alpha(j)}) \geq \beta \cdot B \tag{*}$$

Here  $w_S(x_h)$  is what we call the **natural weight** of point  $x_h$  in  $S$ , which is the maximum number of points in  $S$  whose distances to  $x_h$  sum to at most  $2B$ :

$$w_S(x_h) := \max\{|S'| : S' \subseteq S, \sum_{s \in S'} d(s, x_h) \leq 2B\}.$$

The condition (\*) states that every pair of these  $t$  points is far apart – according to a weighted notion of distance. In turn, the weights used in this notion of distance are the so-called natural weight of each point, which captures the density of its nearby points. Intuitively, if we have  $t$  well-separated points, then not only must any near-optimal solution use  $t$  labels, but such a solution cannot combine the points near  $x_{\alpha(i)}$  and the points near  $x_{\alpha(j)}$  into a single cluster.

**Pivots.** The algorithm is divided into at most  $k$  **phases**. For each phase  $t$ , the algorithm maintains a collection of points  $p_1, \dots, p_t$  from the online stream  $X$  which we call *pivots*. The pivots  $p_1, \dots, p_t$  stay fixed during phase  $t$ . The key property they should satisfy is that they

are well-separated with respect to the points seen so far. Between phases, we increase the number of pivots, thus allowing the algorithm to use more labels. The pivot  $p_i$  is associated with the label  $i$ . Thus, during phase  $t$ , there are  $t$  clusters (i.e.,  $t$  labels in use). When a new point arrives, it is assigned the label  $i$  of the pivot  $p_i$  nearest to it (so we maintain a greedy labelling rule). Importantly, though, the *location* of the pivot  $p_i$  for label  $i$  may change over time. Roughly speaking, this occurs when there is a better representative for cluster  $i$ .

**Pivots vs. centers.** While one might reasonably think that the pivots are intuitively (low-cost) centers for the clusters, this intuition is only partially correct. Part of the subtlety of the algorithm design is that there are in fact scenarios where some pivots are poor centers for the corresponding clusters, but still good representatives for making cluster assignment decisions. What is critical is that the pivots are located so as to guarantee that using greedy assignment in the future results in a relatively low cost assignment; so pivots serve to recruit points to the right clusters. Our algorithm evinces a distinction between a good representative for a cluster in the long-term (a pivot) and a good center at a single moment.

**Invariants.** In order for our cost analysis to be tractable, the algorithm should maintain the following invariants:

- Each pivot  $p_i$  is located in a region where it would not be too costly to assign points arriving there the label  $i$ .
- The pivots  $p_1, \dots, p_t$  for phase  $t$  are well-separated (for some appropriate choice of  $\beta$ ) during phase  $t$ .<sup>1</sup>
- There is no other point that is well-separated from the pivots during a phase. (Otherwise, this indicates that another label can and should be in use.)
- The locations of the pivots should not move very often.

Note that some of these invariants can intuitively be in opposition to each other, which requires that the algorithm design be a bit detailed, as there are several different cases where maintaining this invariant requires different updates to the pivots. We now give an overview of how the algorithm maintains these invariants, highlighting representative cases.

## 2.3 Preliminaries

**Assumptions.** We state our results assuming  $B = \text{OPT}$ , but all still hold by replacing  $\text{OPT}$  with  $B$ .

**Terminology.** Recall from above the *natural weights*  $w_S(\cdot)$ . We will always take  $S$  to be some prefix of the online stream  $X$ . Note  $w_S(p)$  can only increase over time as  $S$  enlarges.

Other terms related to well-separation are: A pair of points  $x_i, x_j$  are  **$\beta$ -attached** with respect to  $w_S$  if  $\min\{w_S(x_i), w_S(x_j)\} \cdot d(x_i, x_j) < \beta \cdot B$ , i.e., the well-separated condition does *not* hold for this pair. A useful way of viewing attachment is that we may move a certain number of points lying near  $x_{\alpha_j}$  to  $x_{\alpha_i}$  at bounded cost (but perhaps not in the reverse direction). We say  $p$  is  **$\beta$ -well-separated from** a set of points  $\{x_{\alpha(1)}, \dots, x_{\alpha(m)}\}$  with respect to  $w_S$  if  $\min\{w_S(p), w_S(x_{\alpha(i)})\} \cdot d(p, x_{\alpha(i)}) \geq \beta \cdot B, \forall i \in [m]$ .

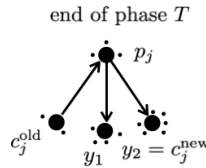
<sup>1</sup>  $\beta$  must be initialized sufficiently large and also decrease as the number of pivots increases.

**Figure notation.** In all figures below, dashed lines indicate well-separation. Solid lines indicate attachment; where included, arrows on solid lines specify the direction of attachment, i.e., point from smaller to larger natural weights (see previous paragraph). Colors except black correspond to cluster labels. Small circles drawn near a larger circle indicate the points attaining the natural weight of the larger point (the maximizing set  $S'$  in the definition of  $w_S$ ).

### 2.4 Subroutines

There are two subroutines used to enlarge the set of pivots from phase  $t$  to phase  $t + 1$ , the **Add Operation** and the **Exchange Operation**. There are subtleties to the execution of these operations that require we also keep track of good centers for the clusters built so far, called **estimated centers**.

**Estimated Centers.** As the pivots are not necessarily good centers (for example, pivot  $p_1$  at location  $-2$  as the points arrive at location  $1$  in Figure 1), the algorithm also maintains a collection  $c_1, \dots, c_T$  of estimated centers for the  $T$  labels<sup>2</sup> that have been used to date. The estimated centers are updated at the end of some phases, and satisfy the invariant that  $c_j$  is a center for label  $j$ 's current cluster with bounded cost. Consider Figure 2. At the start of phase  $T$ ,  $c_j^{\text{old}}$  is the estimated center for points in cluster  $j$ . Points arriving in phase  $T$  that are closer to  $p_j$  than to other pivots are given label  $j$  (by our greedy assignment rule). However, these new points may be concentrated around, for instance,  $y_2$ , so that  $p_j$  is not actually a good center for cluster  $j$  at the end of phase  $T$  (even though it is fine for labelling purposes).



■ **Figure 2** Updating the estimated centers at the end of a phase.

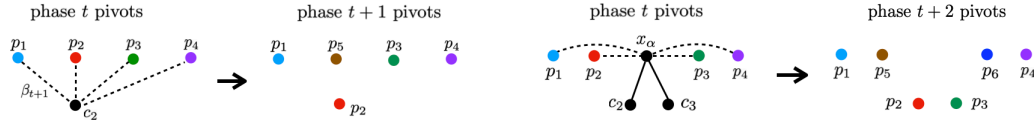
Thus, when it comes time to reset the pivots at the end of phase  $T$ , we might need to move  $p_j$  to the new estimated center  $y_2 = c_j^{\text{new}}$  or to a nearby point. So, estimated centers are not only used in the cost analysis, but critically are used algorithmically to update the locations of pivots. The computation of the estimated centers will involve running an offline approximation algorithm on the points seen to date.

**Add Operation.** An Add Operation is applicable when there is a point  $x_\alpha$  that is well-separated from the current pivots. Intuitively, this means a new label (cluster) can be justified, but the implementation requires the consideration of several possible scenarios. In the simplest scenario  $x_\alpha$  is near a cluster of new points that are all far from previous points, and the pivot  $p_{t+1}$  for the new label  $(t + 1)$  is set to  $x_\alpha$ . In some scenarios an old pivot  $p_i$  ( $i \leq t$ ) is set to  $x_\alpha$  and  $p_{t+1}$  is set to  $p_i$  (so the new pivot location inherits the old label  $i$  and an old pivot location gets the new label  $t + 1$ ). Intuitively, this occurs when the estimated

<sup>2</sup> We use  $T$  instead of  $t$  here to distinguish that this subroutine is only executed at the end of certain phases; see Section 3.

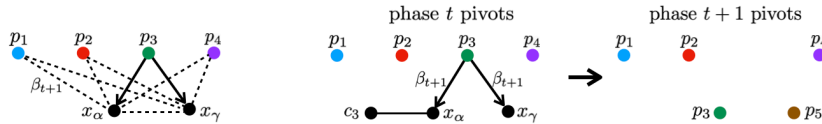


center  $c_i$  for cluster  $i$  is at or near the location of  $x_\alpha$ . See Figure 3 (left); take  $i = 2$ ,  $t = 4$ , and  $x_\alpha = c_2$ . Finally, there are scenarios where  $x_\alpha$  is close to two different clusters; in this case  $x_\alpha$  is never made a pivot and instead *two* pivots are added at the estimated centers of these clusters (so we skip straight to phase  $t + 2$ ). See Figure 3 (right). One must show that this move maintains the well-separation invariant.



■ **Figure 3** Two cases of the Add Operation.

**Exchange Operation.** An Exchange Operation is applicable when there are two points  $x_\alpha$  and  $x_\gamma$  near a pivot  $p_j$  that are well-separated from each other and the other pivots (besides  $p_j$ ). See Figure 4 (left); take  $j = 3$ . So intuitively the cluster of points labeled  $j$  appear to be splitting into two clusters. In the simplest scenario the location of pivot  $p_j$  is set to the location of one of  $x_\alpha$  or  $x_\gamma$ , and the location of the new pivot  $p_{t+1}$  is set to the other. See Figure 4 (right); set  $j = 3$ ,  $t = 4$ .



■ **Figure 4** A case of the Exchange Operation.

This scenario occurs in the instance depicted in Figure 1. The first pivot  $p_1$  is initially set to location  $-2$ . The points arriving at location 1 would all be assigned the label 1 (blue) as there is no point well-separated from  $p_1$  (the points located at 1 are not separated from  $p_1$  because the points at  $p_1$  can be cheaply moved to location 1). When enough points have arrived at the origin, then the points  $x_\alpha = 0$  and at  $x_\gamma = 1$  are near  $p_1$  (because the point at  $p_1$  can be cheaply moved to either  $x_\alpha$  or  $x_\gamma$ ), but are well-separated from each other and the pivots other than  $p_1$ . Thus our algorithm would locate  $p_1$  at 1 and  $p_2$  at the origin. While this gives intuition, there are other more subtle scenarios.

### 3 Algorithm Description

The algorithm sees an online sequence  $X = \{x_1, x_2, \dots, x_n\}$  of points. Let  $X_i = \{x_1, x_2, \dots, x_i\}$ . Let  $w_i$  be shorthand for  $w_{X_i}$ . During any phase  $t$ , the algorithm maintains:

- a collection of previously arriving points  $p_1, \dots, p_t$  that have been designated as *pivots*, where  $t$  is the number of labels used by the algorithm to date and pivot  $p_j$  is associated with label  $j$ ,
- a separation parameter  $\beta_t = 8 \cdot 3^{k-t+2}$ , and
- a collection of previously arriving points  $c_1, \dots, c_s$  ( $s \leq t$ ) that have been designated as estimated centers.

**Phase  $t$**  is the set of time steps when there are  $t$  pivots. Phase 1 is initialized as follows: the first point  $x_1$  is given the label 1, the first pivot  $p_1$  is set to  $x_1$ , and the collection of estimated centers is empty. Let  $T$  be the current phase. The algorithm handles the arrival

of each subsequent point  $x_i$  as follows. It checks whether there is an applicable Add or Exchange Operation, both of which will increase the number of pivots. If so, phase  $T$  ends. First estimated centers  $c_1, \dots, c_T$  are computed, and then, using these, the algorithm carries out consecutive Add and Exchange Operations (giving preference to Add Operations for technical reasons) until there are none left. With each operation, the phase increases and the pivots are reset. Call the last phase in this sequence of consecutive operations  $T^+$ . (Note that  $T^+ \geq T + 1$ .) The point  $x_i$  is the first point labelled during phase  $T^+$  (it is *not* labelled during phase  $T$ ). In summary:

1. **If** there is an applicable Add or Exchange Operation upon the arrival of  $x_i$  **then** compute new Estimated Centers  $c_1, \dots, c_T$ .
  - a. **Repeat** while there is an applicable Add Operation or Exchange Operation.
    - i. **If** there is an applicable Add Operation **then** apply an arbitrary applicable one.
    - ii. **Else** apply an arbitrary applicable Exchange Operation.
2. Give  $x_i$  label  $j$ , where  $p_j$  is the nearest pivot (among  $p_1, \dots, p_{T^+}$ ) to  $x_i$

We then repeat the above steps upon the arrival of  $x_{i+1}$ . Note that if there is  $t$  such that  $T < t < T^+$ , then no points are labelled during phase  $t$ . We call such phases  $t$  during which no points are labelled **intermediate**. During each other phase, at least one point is labelled, and we call such phases **non-intermediate**. So for a non-intermediate phase  $T$ ,  $T^+$  is the first non-intermediate phase after  $T$ , and we will also use  $T^-$  to refer to the last intermediate phase before  $T$ . We now describe the three subroutines.

### 3.1 The Estimated Center Subroutine

This subroutine computes  $T$  new estimated centers  $c_1, \dots, c_T$  from pivots  $p_1, \dots, p_T$ , the points  $X_{i-1}$  that have arrived before  $x_i$ , and estimated centers  $c_1, \dots, c_{T-}$ .

Choose  $y_1, \dots, y_k \in X_{i-1}$  to be an (offline) optimal collection of  $k$  centers<sup>3</sup> for the points in  $X_{i-1}$ . For each **offline optimal center**  $y_h$ ,  $h \in [k]$ , define  $p(y_h)$  to be the pivot with the minimum weighted distance to  $y_h$ , that is,

$$p(y_h) = \arg \min_{p_j} (\min\{w_{i-1}(p_j), w_{i-1}(y_h)\} \cdot d(p_j, y_h)) \quad (\dagger)$$

Say that  $y_h$  is *assigned to*  $p_j$  if  $p(y_h) = p_j$ . For each pivot  $p_j$ , we define the set  $\delta(p_j)$  to contain a subset of the offline optimal centers that are assigned to  $p_j$ , and possibly  $c_j$  as well; the points in  $\delta(p_j)$  are “close” to  $p_j$  in some sense. In particular,  $y_h \in \delta(p_j)$  if  $p(y_h) = p_j$  and  $w_{i-1}(y_h) > w_{i-1}(p(y_h))$ . Also,  $c_j$  is in  $\delta(p_j)$  if  $w_{i-1}(c_j) > w_{i-1}(p_j)$  and  $c_j$  is  $\beta_{t+1}$ -attached to  $p_j$  w.r.t.  $w_{i-1}$ .

As an example, see **Figure 2**. Here,  $\delta(p_j) = \{y_1, y_2\}$ , so  $c_j$  (denoted  $c_j^{\text{old}}$ ) is not in  $\delta(p_j)$ , because the arrow from  $c_j^{\text{old}}$  to  $p_j$  (representing attachment) points in the wrong direction.

For each  $j \in [T]$ , we now define the new **estimated center**  $c_j$ : If  $w_{i-1}(p_j) \geq \max_{p \in \delta(p_j)} w_{i-1}(p)$  then  $c_j = p_j$ , else

$$c_j = \arg \max_{p \in \delta(p_j)} w_{i-1}(p) \quad (\ddagger)$$

So in **Figure 2**,  $c_j$  is updated to  $y_2$  (denoted  $c_j^{\text{new}}$ ), because  $y_2$  has the largest weight in  $\delta(p_j)$ . Intuitively, this means that  $c_j^{\text{new}}$  is now a better center for cluster  $j$  than, say,  $c_j^{\text{old}}$ .

<sup>3</sup> To run in poly-time, replace with any constant approximation algorithm. This algorithm’s cost will change by a constant factor.

### 3.2 The Add Operation Subroutine

Let  $t \geq T$  be the number of pivots when an Add Operation is called (during an execution of (i) above). The Add Operation applies if there is a point  $x_\alpha \in X_i$  such that  $x_\alpha$  is  $\beta_{t+1}$ -well-separated from the current pivots  $p_1, \dots, p_t$  with respect to the weights  $w_i$ . (**E.g., Figure 3, left, with  $t = 4$ .**) The Add Operation depends on  $x_\alpha$ ,  $X_i$ , the current pivots  $p_1, \dots, p_t$ , and estimated centers  $c_1, \dots, c_T$ . In most cases, the Add Operation adds  $x_\alpha$  to the set of pivots, and changes the location of up to two previous pivots (**Figure 3**).

Define  $w_t := w_{i-1}$  if  $t = T$  and  $w_t := w_i$  if  $t > T$ .<sup>4</sup>

1. **If** there is an estimated center  $c_j$  that is  $\beta_{t+1}$ -well-separated from  $p_1, \dots, p_t$  w.r.t.  $w_i$  then set  $p_{t+1} = p_j$  and set  $p_j = c_j$ . (**Figure 3, left**)
2. **Else if** it is the case that for every estimated center  $c_j$  that is  $\beta_{t+2}$ -attached to  $x_\alpha$  w.r.t.  $w_i$  it is also the case that  $w_t(c_j) < w_t(p_j)$ , then set  $p_{t+1} = x_\alpha$ .
3. **Else if** there exists a unique estimated center  $c_j$  is  $\beta_{t+2}$ -attached to  $x_\alpha$  w.r.t.  $w_i$  and  $w_t(c_j) \geq w_t(p_j)$  then set  $p_{t+1} = p_j$  and  $p_j = x_\alpha$ .
4. **Else** Let  $c_f$  and  $c_g$  be estimated centers such that each is  $\beta_{t+2}$ -attached to  $x_\alpha$  w.r.t.  $w_i$ ,  $w_t(c_f) \geq w_t(p_f)$ , and  $w_t(c_g) \geq w_t(p_g)$ . Set  $p_{t+1} = p_f$ ,  $p_{t+2} = p_g$ ,  $p_f = c_f$ , and  $p_g = c_g$ . (**Figure 3, right**)

Note in the last case that we skip to phase  $t + 2$ .

### 3.3 The Exchange Operation Subroutine

The Exchange Operation subroutine is applicable if there exists two points  $x_\alpha$  and  $x_\gamma$  in  $X_i$ , and a pivot  $p_j$  such that:

- $x_\alpha$  and  $x_\gamma$  are each  $\beta_{t+1}$ -attached to  $p_j$  w.r.t.  $w_i$ ,
- $w_i(p_j) \leq w_i(x_\alpha)$ ,
- $w_i(p_j) \leq w_i(x_\gamma)$ , and
- The collection of the  $t + 1$  points, consisting of  $x_\alpha$ ,  $x_\gamma$ , and the pivots other than  $p_j$ , are  $\beta_{t+1}$ -well-separated w.r.t.  $w_i$ . (**E.g., Figure 4, left, with  $t = 4$ .**)

The Exchange Operation depends on  $x_\alpha$ ,  $x_\gamma$ ,  $X_i$ , the current pivots  $p_1, \dots, p_t$ , and estimated centers  $c_1, \dots, c_T$ . In most cases, the Exchange Operation adds  $x_\alpha$  and  $x_\gamma$  to and deletes  $p_j$  from the set of pivots, and possibly changes the location of one previous pivot.

1. **If**  $j > T$  then set  $p_j = x_\alpha$  and  $p_{t+1} = x_\gamma$ .
2. **Else if**  $w_i(c_j) < w_i(p_j)$  then set  $p_j = x_\alpha$  and  $p_{t+1} = x_\gamma$ .
3. **Else if**  $c_j$  is  $\beta_{t+2}$ -attached to  $x_\alpha$  w.r.t.  $w_i$  then set  $p_j = x_\alpha$  and  $p_{t+1} = x_\gamma$ . (**Figure 4, right**)
4. **Else if**  $c_j$  is  $\beta_{t+2}$ -attached to  $x_\gamma$  w.r.t.  $w_i$  then set  $p_j = x_\gamma$  and  $p_{t+1} = x_\alpha$ .
5. **Else** set  $p_{t+1} = x_\alpha$ ,  $p_{t+2} = x_\gamma$ , and  $p_j = c_j$ .

## 4 Algorithm Invariants and Analysis

In this section, we state the key technical lemmas. We defer full proofs to the Appendix.

► **Theorem 3.** *The algorithm uses at most  $k$  labels.*

► **Theorem 4.** *The algorithm's cost is  $O(k^5 \cdot 3^k \cdot OPT)$ .*

<sup>4</sup> We are overloading subscripts here for ease. We could instead write  $v_t$ , but we retain  $w$  to recall weights.

## 4.1 Notation

- $p_1^t, \dots, p_t^t$  denote the pivots for labels 1 through  $t$ , respectively, during phase  $t$ .
- $w^t$  are the natural weights at the end of phase  $t$ .
- $X(t)$  is the set of points assigned a label before or during phase  $t$ .
- For  $j \in [t]$ ,  $C_j^t$  is the set of points labelled  $j$  in phases 1 through  $t$ .
- For  $j \in [T]$ ,  $c_j^T$  is the estimated center (‡) computed at end of non-intermediate phase  $T$ .
- $y_1^T, \dots, y_k^T$  are the offline optimal centers computed at the end of phase  $T$  in The Estimated Center Subroutine.
- $P_T = \{p_1^T, \dots, p_T^T, y_1^T, \dots, y_k^T\}$
- $\text{cost}(S; c) = \sum_{p \in S} d(p, c)$  for  $S \subseteq X$  and  $c \in X$ .

## 4.2 Invariants

In the next two lemmas, we show that our algorithm maintains two key invariants. The full proofs are deferred to the full version, although we give a proof sketch of Lemma 5 in the appendix.

► **Lemma 5.** *Let  $t \in [k]$ . The algorithm maintains the invariant that  $p_1^t, \dots, p_t^t$  are  $\beta_t$ -well-separated w.r.t. the natural weights at the start of phase  $t$  (and after).*

Lemma 5 directly implies Theorem 3 once we show that there can be no more than  $k$  well-separated points in  $X$  and note that we have set  $\beta_1$  sufficiently large.

Next is a key technical lemma. It states that the estimated center  $c_j^{T^-}$  for the points given label  $j$  *before* phase  $T$  is close, in a weighted sense, to the pivot for label  $j$  in phase  $T$ . This is key to showing that points in cluster  $j$  that are labelled *before* phase  $T$  can be combined with those that are labelled *during* phase  $T$  at bounded cost. This lemma is in tension with the prior one because a pivot must be placed in a location where it is both well-separated from other pivots and is close to previously arriving points in its cluster.

► **Lemma 6.** *Let  $T$  be a non-intermediate phase and let  $j \in [T]$ . If  $T > 1$ , at least one of the following holds:*

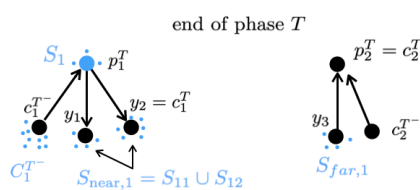
- (a)  $w^{T^-}(c_j^{T^-}) \leq w^T(p_j^T)$  and  $w^{T^-}(c_j^{T^-}) \cdot d(c_j^{T^-}, p_j^T) \leq \beta_{T^-}(T - T^-) \cdot \text{OPT}$ .
- (b)  $c_j^{T^-}$  is  $\beta_{T+1}$ -attached to  $p_j^T$  w.r.t.  $w^T$ .

## 5 Bounding Cost

We show by induction that the estimated center  $c_j^T$  is good for *all* points given label  $j$  by the end of phase  $T$ . Taking  $T$  to be the last phase gives our main result. We follow Figure 5. By definition of attached, a certain number of points sitting at the head (equal to the head's natural weight) of an arc can be moved to the tail at bounded cost. First we address the cost of  $C_1^{T^-}$ , the points given label 1 before phase  $T$ . We inductively assume these can be moved to  $c_1^{T^-}$  at bounded cost. From there, we need to move them to  $c_1^T$  at bounded cost. This can be done by showing that (1)  $|C_1^{T^-}|$  is a bounded factor away from the natural weight of  $c_1^{T^-}$  (Lemma 8), and (2)  $c_1^T$  is “close” to  $p_1^T$  (Lemma 8). Together these imply that we can move the points in  $C_1^{T^-}$  along the arc from  $c_1^{T^-}$  to  $p_1^T$  at bounded cost.

Next we show that the points given label 1 *during* phase  $T$ , call them  $C_1$ , can also be moved to  $c_1^T$  at bounded cost. This is where we use the set of offline optimal centers  $y_1^T, \dots, y_k^T$  computed during the Estimated Centers Subroutine. Importantly, since during a phase every point is attached to at least one pivot (otherwise we execute an Add Operation

and leave the phase), each offline center  $y_i^T$  is attached to a pivot. We partition the points in  $C_1$  based on which center  $y_i^T$  they are assigned to in the *offline* optimal solution. The set of points in  $C_1$  that are assigned to centers attached to the pivot for label 1,  $p_1^T$ , is called  $S_{near,1}$ . In Figure 5, these are points assigned to  $y_1^T$  and  $y_2^T$ . One can show, using that during a phase no Exchange Operation occurs, that these can be moved to  $c_1^T$  at bounded cost. The set of points that are assigned to centers that are attached to a pivot for a different label, say label 2, is called  $S_{far,1}$ . These points are misclassified in the sense that the online and offline algorithms classify them differently. However, we show their cost is still controlled. Specifically, the well-separated invariant implies that (1) these points can be moved to  $p_1^T$  at bounded cost, and (2) the number of them is a bounded factor away from the natural weight of  $p_1^T$  (Lemma 7). These two properties imply we can move the points in  $S_{far,1}$  to  $p_1^T$ , and then to  $c_1^T$ , at bounded cost.



■ **Figure 5** The points given label 1 (blue) before or during phase  $T$  are partitioned as in the text.

► **Lemma 7.** *Let  $T$  be a non-intermediate phase. For any  $j \in [T]$ , let  $C_j$  be the points given label  $j$  during phase  $T$ , i.e.,  $C_j = C_j^T \setminus C_j^{T-}$ . Define  $S_{ji}$  to be the set of elements in  $C_j$  assigned to  $y_i$  in the clustering of  $X(T) \setminus X(T^-)$  induced by  $P_T$ . Define  $S_{far,j} = \bigcup_{i:p(y_i) \neq p_j^T} S_{ji}$ . Then*

1.  $cost(S_{far,j}; p_j^T) \leq k \cdot (\beta_{T+1} + 2) \cdot OPT$ , and
2.  $|S_{far,j}| \leq k \cdot w^T(p_j^T)$ , where  $w^T$  denotes the natural weights at the end of phase  $T$ .

► **Lemma 8.** *Let  $T$  be a non-intermediate phase and  $j \in [T]$ . Let  $w^T(c_j^T)$  be the natural weight of  $c_j^T$  at the end of phase  $T$  and  $C_j^T$  be the set of points in cluster  $j$  by the end of phase  $T$ . Then  $|C_j^T| \leq (2k + 1) \cdot T \cdot w^T(c_j^T)$ .*

The final lemma below shows that the cost of our algorithm's solution at the end of phase  $T$  is bounded against  $OPT$ . Taking  $T$  to be the last phase gives Theorem 4.

► **Lemma 9.** *Let  $T$  be a non-intermediate phase and  $j \in [T]$ . Then  $cost(C_j^T)$  is bounded against center  $c_j^T$ , i.e.,  $\sum_{x \in C_j^T} d(x, c_j^T) \leq g(T, k) \cdot OPT$ , where*

$$g(T, k) = T \cdot g(k) \quad \text{and} \quad g(k) = \beta_1(2k^3 + 3k^2 + 5k + 1) + 2k + 4.$$

## 6 Conclusion

This paper gives the first online algorithm for **cluster-based**  $k$ -median clustering, with competitive ratio independent of  $n$  and  $\Delta$ , that does not recluster or use additional centers. We take a learning-augmented approach, assuming minimal a priori information in the form of an upper bound  $B$  on the optimal cost. Prior to this work, it was not known that any algorithm could have bounded worst-case guarantees. Interestingly, we remark that if the algorithm does not know  $B$  and reclustering is allowed, our results imply an algorithm that maintains a solution competitive against the optimal solution on the points that have arrived *so far*. Reclustering an  $O(\log(n\Delta))$  number of times, the algorithm is always  $O(1)$ -competitive when  $k$  is a constant at each point in time. This matches the number of reclusterings used in prior work for the consistent center case.

## References

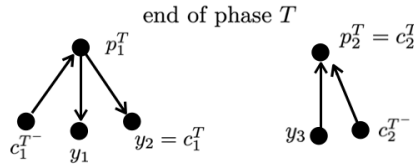
- 1 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for  $k$ -median and facility location problems. *SIAM Journal of Computing*, 33(3):544–562, 2004.
- 2 Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable  $k$ -means++. *Proc. VLDB Endow.*, 5(7):622–633, 2012.
- 3 Robi Bhattacharjee, Jacob Imola, Michal Moshkovitz, and Sanjoy Dasgupta. Online  $k$ -means clustering on arbitrary data streams. In *International Conference on Algorithmic Learning Theory*, pages 204–236. PMLR, 2023.
- 4 Robi Bhattacharjee and Michal Moshkovitz. No-substitution  $k$ -means clustering with adversarial order. In *Algorithmic Learning Theory*, pages 345–366. PMLR, 2021.
- 5 Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.
- 6 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 Transactions on Algorithms*, 13(2):23:1–23:31, 2017.
- 7 T-H. Hubert Chan, Arnaud Guerqin, and Mauro Sozio. Fully dynamic  $k$ -center clustering. In *World Wide Web Conference*, pages 579–587, 2018.
- 8 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal of Computing*, 33(6):1417–1440, 2004.
- 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 Systems Sciences*, 65(1):129–149, 2002.
- 10 Vincent Cohen-Addad, Niklas Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully dynamic consistent facility location. In *Conference on Neural Information Processing Systems*, pages 3250–3260, 2019.
- 11 Hendrik Fichtenberger, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Consistent  $k$ -clustering for general metrics. In *ACM-SIAM Symposium on Discrete Algorithms*, 2021.
- 12 Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian. Consistent  $k$ -median: Simpler, better and robust. In *International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1135–1143, 2021.
- 13 K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and  $k$ -median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.
- 14 Silvio Lattanzi and Sergei Vassilvitskii. Consistent  $k$ -clustering. In Doina Precup and Yee Whye Teh, editors, *International Conference on Machine Learning*, pages 1975–1984, 2017.
- 15 Shi Li and Ola Svensson. Approximating  $k$ -median via pseudo-approximation. *SIAM Journal of Computing*, 45(2):530–547, 2016.
- 16 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko.  $k$ -means clustering. talk slides.
- 17 Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online  $k$ -means clustering. In *Workshop on Algorithm Engineering and Experiments*, pages 81–89, 2016.
- 18 A. Meyerson. Online facility location. In *IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.
- 19 Adam Meyerson, Liadan O’Callaghan, and Serge Plotkin. A  $k$ -median algorithm with running time independent of data size. *Machine Learning*, 56(1):61–87, 2004.
- 20 Benjamin Moseley, Heather Newman, and Kirk Pruhs. Online  $k$ -median with consistent clusters. *arXiv preprint*, 2023. [arXiv:2303.15379](https://arxiv.org/abs/2303.15379).
- 21 Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021.

The Appendix is organized as follows. In Appendix A, we introduce some additional terminology and notation. In Appendix B, we introduce a few propositions that will be useful for proving the main lemmas. In Appendix C, we prove Lemma 5, the well-separation invariant; this is a rather involved proof, so we include both a proof sketch with the high-level ideas, and defer the full proof to the full version. Then, in Appendix D, we show how Lemma 5 swiftly implies Theorem 3. In the remaining appendices, we prove the lemmas in Section 5. The last lemma, Lemma 9, directly implies Theorem 4.

## A Terminology

Throughout this appendix, we take  $B = \text{OPT}$  for simplicity. Our results still hold as long as  $B \geq \text{OPT}$ . Below is some additional terminology used in the proofs.

- Let  $y_1^T, \dots, y_k^T$  be the optimal collection of  $k$  centers computed at the end of phase  $T$  in The Estimated Center Subroutine. Let  $P_T = \{p_1^T, \dots, p_T^T, y_1^T, \dots, y_k^T\}$  and call this set the **offline centers** for phase  $T$ . When the context is clear, we may omit the superscript  $T$  in  $y_i^T$ .
- The **attachment digraph**  $D(T)$  is a bipartite digraph with vertex set  $P_T$ , plus  $\{c_j^{T-} \mid j \leq T-1\}$  if  $T > 1$ , partitioned as  $(\{p_1^T, \dots, p_T^T\}, \{y_1, \dots, y_k, c_1^{T-}, \dots, c_{T-1}^{T-}\})$ . There is a directed arc  $(y_i, p(y_i))$  if  $w^T(y_i) \leq w^T(p(y_i))$  and a directed arc  $(y_i, p(y_i))$  otherwise. If  $c_j^{T-}$  and  $p_j^T$  are  $\beta_{T+1}$ -attached w.r.t.  $w^T$ , add the arc  $(c_j^{T-}, p_j^T)$  if  $w^T(c_j^{T-}) \leq w^T(p_j^T)$  and the arc  $(p_j^T, c_j^{T-})$  otherwise.  $\delta^+(p_j^T)$  and  $\delta^-(p_j^T)$  denote the out- and in-degree of  $p_j^T$ . See Figure 6.



■ **Figure 6** The attachment digraph  $D(T)$ . Arrows represent attached pairs, and arrows point from smaller to larger natural weights. So we may move a certain number of points near the head, to the tail at bounded cost.

## B Helper Propositions

In this section, we present a few short propositions that will be useful in the remaining proofs. All excluded proofs are deferred to the full version.

The following fact justifies that the offline optimal centers  $y_1^T, \dots, y_k^T$  have cost at most  $2\text{OPT}$  on the points that arrive during phase  $T$ . This is used at various points in the analysis.

► **Fact 10** (Fact 2.1 in [19]). *Let  $N$  be a set of points, with  $S \subseteq N$ . Let  $k$  be an integer with  $0 \leq k \leq n$ . Let  $K \subseteq S$  be the  $k$ -element subset of  $S$  minimizing  $\sum_{x \in S} d(x, K)$  where  $d(\cdot, \cdot)$  is the distance function on  $N$ , and  $d(x, K)$  denotes  $\min_{m \in K} d(x, m)$ . Then if  $K'$  is a  $k$ -element subset of  $N$ ,  $\sum_{x \in S} d(x, K) \leq 2 \sum_{x \in S} d(x, K')$ .*

The following proposition is a weighted version of the triangle inequality.

► **Proposition 11.** *Let  $x, y, p$  be three points in some set  $S$ , and let  $w : S \rightarrow \mathbb{Z}_+$  be a weight function on  $S$ . Assume that  $\beta, \beta_x, \beta_y, B > 0$ . Suppose that  $w(x) \leq w(p)$  and that  $x$  and  $y$  are  $\beta$ -well-separated w.r.t.  $w$ . If  $x$  and  $p$  are  $\beta_x$ -attached w.r.t.  $w$ , and  $y$  and  $p$  are  $\beta_y$ -attached w.r.t.  $w$ , then  $\beta < \beta_x + \beta_y$ .*

Next, we show that a point set cannot contain more than  $k$  pairwise  $\beta$ -well-separated points for  $\beta$  a sufficiently large constant. This allows us to bound the number of labels used.

► **Proposition 12.** *Let  $X$  be a set of points whose optimal  $k$ -median cost using  $k$  centers is  $OPT$ . Let  $\{x_1, \dots, x_l\}$  be a set of points in  $X$ , and let  $w_X$  denote their natural weights in  $X$ . Let  $\beta > 8$ . If  $\{x_1, \dots, x_l\}$  is  $\beta$ -well-separated w.r.t.  $w_X$ , then  $l \leq k$ .*

The next two propositions will be used to aid the proofs of Lemmas 5 and 6. Recall that for each non-intermediate phase  $T$ , we defined a set of offline centers  $P_T$  that has cost at most  $2OPT$  on  $X(T)$  (Appendix A and Fact 10). In order to compare the (low-cost) offline clustering induced by  $P_T$  to our online algorithm's clustering, we relate the offline set of centers  $P_T$  (which we *know* have bounded cost on  $X(T)$ ) to the pivots in phase  $T$  (which are used to make the greedy online choices) in the next proposition.

► **Proposition 13.** *Let  $P_T = \{p_1^T, \dots, p_T^T, y_1, \dots, y_k\}$  be as in Appendix A. Then  $y_i$  and  $p(y_i)$  are  $\beta_{T+1}$ -attached w.r.t. the natural weights  $w^T$  at the end of phase  $T$ .*

Each attached pair in Proposition 13 is encoded in the digraph  $D(T)$  by a directed arc. So, we can now think of this directed arc as representing the direction in which we could move a certain number of points sitting near one endpoint to the other at bounded cost.

Next we show that the estimated center for a cluster at the end of a phase is attached to the pivot for that cluster in that phase. Thus, while the pivot itself may not be a good center for the cluster, the pivot is close to the estimated center (in at least one direction, in a weighted sense).

► **Proposition 14.** *The estimated center  $c_j^T$  is  $\beta_{T+1}$ -attached to  $p_j^T$  w.r.t. the natural weights  $w^T$  at the end of phase  $T$ . Further,  $w^T(c_j^T) \geq w^T(p_j^T)$ , with equality if and only if  $c_j^T = p_j^T$ .*

## C Proof of Lemma 5

As the proof of Lemma 5 is a rather involved double induction, we provide a proof sketch which pulls out the hard cases. In the full version, we give the full proof.

**Proof sketch of Lemma 5.** The proof is by induction. However, we need to couple the induction with a statement about the relative position of the estimated center for a cluster (which stays fixed between intermediate phases) to that cluster's pivot, which may change often as we consecutively reset the pivots between intermediate phases. Roughly, we prove below that if the estimated center for cluster  $j$  has not separated entirely from the present set of pivots, then it must be close (in a weighted sense) to the present pivot for label  $j$ .

▷ **Claim 1.** Let  $w_{i-1}$ ,  $w_i$ , and  $w_t$  be as Section 3.2. For each  $j \in [T]$  and  $t \in [T, T^+]$  such that  $p_1^t, \dots, p_t^t$  are defined,<sup>5</sup>

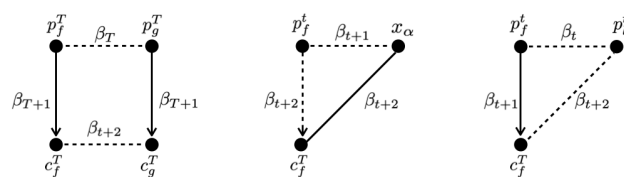
$$p_1^t, \dots, p_t^t \text{ are } \beta_t\text{-well-separated w.r.t. } w_t. \quad (\diamond)$$

Moreover, at least one of the following properties holds:

- (a)  $c_j^T$  is  $\beta_{t+1}$ -well-separated from  $p_1^t, \dots, p_t^t$  w.r.t.  $w_i$ .
- (b)  $c_j^T$  is  $\beta_{t+1}$ -attached to  $p_j^t$  w.r.t.  $w_t$ .
- (c)  $c_j^T$  is  $f(t, T)$ -attached to  $p_j^t$  w.r.t.  $w_t$  and  $w_t(c_j^T) < w_t(p_j^t)$ , where  $f(t, T) = \beta_T \cdot (t - T)$ .

<sup>5</sup> Recall in Case 4 of the Add Operation and Case 5 of the Exchange Operation, we go directly from  $t$  to  $t + 2$  pivots, skipping phase  $t + 1$ .





■ **Figure 7** Cases (i) –(iii) in the proof sketch of Lemma 5. Dashed lines indicate well-separation and solid lines indicate attachment, labelled with the appropriate parameters. Arrows go from smaller to larger natural weights.

For the proof sketch we focus on Case 4 of the Add Operation, which will give a flavor of the arguments. This is a concerning case a priori; for, if we were to add  $x_\alpha$  to the set of pivots as in Cases 2 and 3, it is ambiguous as to whether  $x_\alpha$  should be associated with label  $f$  or  $g$ , as both  $c_f^T$  and  $c_g^T$  are close to  $x_\alpha$ . We maneuver around the issue by making  $c_f^T$  and  $c_g^T$  new pivots and excluding  $x_\alpha$ . However, it is not immediately clear that such a step will preserve the desired invariants. To give intuition, we suppress the separation parameters and the precise weights used, though emphasize both are brittle (e.g., the arguments rely heavily on  $\beta_t$  decreasing with  $t$ ). The directions of attachment between points (arrows in Figure 7) are also crucial. We will also see why we need to couple the induction with (a) –(c).

To prove the inductive step for  $(\diamond)$  when Case 4 of the Add Operation is performed, we need to show (i)  $c_f^T$  and  $c_g^T$  are well-separated, (ii), WLOG,  $c_f^T$  is well-separated from  $p_f^t$ , and (iii), WLOG,  $c_f^T$  is well-separated from  $p_l^t$ ,  $l \neq f$ . See Figure 7. When we say “close” or “far” below, we always mean in a weighted sense. For (i), because  $p_f^T$  is close to  $c_f^T$  (Proposition 14) and likewise for  $p_g^T, c_g^T$ , then  $c_f^T$  and  $c_g^T$  cannot be close, since this would violate that  $p_f^T$  and  $p_g^T$  are (inductively) far. To prove (ii), note  $x_\alpha$  is far from  $p_f^t$  by assumption of the Add Operation, and  $c_f^T$  is close to  $x_\alpha$  by assumption of Case 4, so  $p_f^t$  and  $c_f^T$  must be far. Finally for (iii), one can (inductively) deduce that (b) must hold when  $j = f$ , so  $c_f^T$  and  $p_f^t$  are close; but, since  $p_f^t$  and  $p_l^t$  are (inductively) far,  $c_f^T$  and  $p_l^t$  must be far.

Proving the inductive step for (a) –(c) involves detailed casework. The Add and Exchange Operations are engineered so that, loosely speaking, an estimated center is either attached to the corresponding present pivot, or else breaks off to form its own pivot. A main subtlety is the direction and strength of attachment, e.g., property (c). Another is the sequence of operations, specifically, the Add Operation taking precedence over the Exchange Operation. ◀

## D Proof of Theorem 3

**Proof of Theorem 3.** The number of labels used by the algorithm is the number of pivots in the last phase. By Lemma 5, we maintain the invariant that pivots  $p_1^t, \dots, p_k^t$  are  $\beta_t$ -well-separated w.r.t. the natural weights at every time step in phase  $t$ . Suppose to the contrary that the final number of pivots is strictly more than  $k$ . Then at some point there are  $t = k + 1$  or  $t = k + 2$  pivots<sup>6</sup> that are  $\beta_t$ -well-separated w.r.t. the natural weights throughout phase  $t$ . But  $\beta_{k+2} = 8$ , and it is impossible for  $k + 2$  points to be 8-well-separated, by Proposition 12. We conclude the final number of pivots is at most  $k$ . ◀

<sup>6</sup> The algorithm may skip a phase, hence we consider both cases.

## E Proof of Lemma 7

**Proof of Lemma 7.** WLOG, let  $j = T$ . For  $c \in P_T$ , let  $m(c)$  be the number of points assigned to  $c$  in the clustering of  $X(T) \setminus X(T^-)$  induced by the centers  $P_T$ , i.e., in this clustering every point is assigned to the *nearest* point in  $P_T$ . For shorthand, let  $w$  denote the natural weights  $w^T$  of points at the end of phase  $T$ .

► **Observation 2.** For  $c \in P_T$ ,  $w(c) \geq m(c)$ .

This follows from the definition of  $w(c)$  and the fact that there are  $m(c)$  points whose movement cost to  $c$  is at most  $2\text{OPT}$ , by construction of  $P_T$ .

► **Observation 3.** If  $(p(y_i), y_i)$  is a directed edge in  $D(T)$ , then  $w(p(y_i)) \cdot d(p(y_i), y_i) < \beta_{T+1} \cdot \text{OPT}$ . Likewise, if  $(y_i, p(y_i))$  is a directed edge in  $D(T)$ , then  $w(y_i) \cdot d(p(y_i), y_i) < \beta_{T+1} \cdot \text{OPT}$ .

This follows from the definition of  $D(T)$  and Proposition 13.

Call the points in  $S_{far,T}$  far points. In the claims below, we show that the far points can be moved to  $p_T^T$  at bounded cost (Claims 1 and 2), and that there are not too many far points relative to the weight of  $p_T^T$  (Claim 3). In turn, we will be able to *charge* the cost of the far points to  $p_T^T$ .

▷ **Claim 1.** Let  $p(y_i) \neq p_T^T$ . Suppose  $w(y_i) > w(p(y_i))$ . Then  $\text{cost}(S_{T_i}; p_T^T) \leq (\beta_{T+1} + 2)\text{OPT}$ .

Proof. WLOG, let  $p(y_i) = p_1^T$ . We consider two cases.

► **Case 1.**  $|S_{T_i}| \geq w(p_1^T)$ . We will show this case cannot happen.

We know that  $w(y_i) \geq m(y_i) \geq |S_{T_i}| \geq w(p_1^T)$ , and by Observation 3, that  $w(p_1^T) \cdot d(p_1^T, y_i) < \beta_{T+1} \cdot \text{OPT}$ . By Proposition 11, this implies  $w(p_1^T) \cdot d(y_i, p_1^T) \geq 2\beta_{T+1} \cdot \text{OPT}$ .

Since  $|S_{T_i}| \geq w(p_1^T)$ , there exists  $S'_{T_i} \subseteq S_{T_i}$  such that  $|S'_{T_i}| = w(p_1^T)$ . In turn,  $\text{cost}(S'_{T_i}; p_1^T) \leq \text{cost}(S'_{T_i}; y_i) + w(p_1^T) \cdot d(y_i, p_1^T) < (\beta_{T+1} + 2) \cdot \text{OPT}$ , since  $P_T$  is a clustering with cost at most  $2\text{OPT}$ . On the other hand,

$$\text{cost}(S'_{T_i}; p_1^T) \geq \sum_{p \in S'_{T_i}} d(y_i, p_1^T) - \sum_{p \in S'_{T_i}} d(p, y_i) = w(p_1^T) \cdot d(y_i, p_1^T) - \sum_{p \in S'_{T_i}} d(p, y_i) \geq (2\beta_{T+1} - 2)\text{OPT}.$$

Since  $\beta_{T+1} \geq 4$ ,  $\beta_{T+1} + 2 \leq 2\beta_{T+1} - 2$ , so  $\text{cost}(S'_{T_i}; p_1^T) < \text{cost}(S'_{T_i}; p_1^T)$ , which violates that  $T = \arg \min_{j \in [T]} d(p, p_j^T)$  for all  $p \in S'_{T_i} \subseteq C_T$ .

► **Case 2.**  $|S_{T_i}| \leq w_t(p_1^T)$ .

In this case, we know that since  $w(p_1^T) \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \text{OPT}$ , we also have  $|S_{T_i}| \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \text{OPT}$ . By the triangle inequality,

$$\text{cost}(S_{T_i}; p_1^T) \leq \text{cost}(S_{T_i}; y_i) + |S_{T_i}| \cdot d(y_i, p_1^T) \leq 2\text{OPT} + \beta_{T+1} \cdot \text{OPT}.$$

Since  $\text{cost}(S_{T_i}; p_T^T) \leq \text{cost}(S_{T_i}; p_1^T)$  by the greedy procedure, this proves Claim 1. ◁

▷ **Claim 2.** Let  $p(y_i) \neq p_T^T$ . Suppose that  $w(y_i) \leq w(p(y_i))$ . Then  $\text{cost}(S_{T_i}; p_T^T) \leq (\beta_{T+1} + 1)\text{OPT}$ .

Proof. WLOG, let  $p(y_i) = p_1^T$ . By Observation 3,  $w(y_i) \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \text{OPT}$ . Further,  $|S_{T_i}| \leq m(y_i) \leq w(y_i)$ , so  $|S_{T_i}| \cdot d(y_i, p_1^T) < \beta_{T+1} \cdot \text{OPT}$ . So:

$$\text{cost}(S_{T_i}; p_T^T) \leq \text{cost}(S_{T_i}; p_1^T) \leq \text{cost}(S_{T_i}; y_i) + |S_{T_i}| \cdot d(y_i, p_1^T) \leq 2\text{OPT} + \beta_{T+1} \cdot \text{OPT}. \triangleleft$$

▷ **Claim 3.** Let  $p(y_i) \neq p_T^T$ . Then  $|S_{T_i}| \leq w(p_T^T)$ .

Proof. As before, assume WLOG that  $p(y_i) = p_1^T$ .

► **Case 1.**  $w(y_i) > w(p_1^T)$ .

We know from the proof of Claim 1, Case 1 that this implies  $|S_{T_i}| < w(p_1^T)$ . We have

$$\begin{aligned} |S_{T_i}| \cdot d(p_T^T, y_i) &= \sum_{p \in S_{T_i}} d(y_i, p_T^T) \leq \sum_{p \in S_{T_i}} d(p, p_T^T) + \sum_{p \in S_{T_i}} d(p, y_i) \\ &\leq (\beta_{T+1} + 2)\text{OPT} + 2\text{OPT} && \text{(Claim 1)} \\ &\leq 2\beta_{T+1} \cdot \text{OPT} \leq w(p_T^T) \cdot d(p_T^T, y_i) \end{aligned}$$

where in the last line we have applied Proposition 11, using that  $w(y_i) > w(p_1^T)$ , Observation 3, and  $p_1^T$  and  $p_T^T$  are  $\beta_T$ -well-separated w.r.t.  $w$ . Finally, dividing both ends of the chain of inequalities by  $d(p_T^T, y_i)$  gives  $|S_{T_i}| \leq w(p_T^T)$ , as desired.

► **Case 2.**  $w(y_i) \leq w(p_1^T)$ .

Consider when  $w(p_T^T) \geq w(y_i)$ . Then  $w(p_T^T) \geq w(y_i) \geq m(y_i) \geq |S_{T_i}|$ , so the claim follows.

So the last case to consider is when  $w(p_T^T) < w(y_i)$ . It suffices to show that  $w(p_T^T) \cdot d(p_T^T, y_i) \geq 2\beta_{T+1} \cdot \text{OPT}$ ; then, we can just apply the argument in Case 1. Suppose to the contrary that  $w(p_T^T) \cdot d(p_T^T, y_i) < 2\beta_{T+1} \cdot \text{OPT}$ . Then

$$\begin{aligned} \beta_T \cdot \text{OPT} &\leq w(p_T^T) \cdot d(p_T^T, p_1^T) \leq w(p_T^T) \cdot d(p_T^T, y_i) + w(p_T^T) \cdot d(y_i, p_1^T) \\ &\leq 2\beta_{T+1} \cdot \text{OPT} + w(p_T^T) \cdot d(y_i, p_1^T) \\ &< 2\beta_{T+1} \cdot \text{OPT} + w(y_i) \cdot d(y_i, p_1^T) \\ &< 2\beta_{T+1} \cdot \text{OPT} + \beta_{T+1} \cdot \text{OPT} = \beta_T \cdot \text{OPT} \end{aligned}$$

where the second-to-last line follows from Observation 3. The left-hand and right-hand sides give a contradiction, concluding the proof of the case and the claim. ◁

▷ **Claim 4.**  $\text{cost}(S_{far,T}; p_T^T) \leq k \cdot (\beta_{T+1} + 2)\text{OPT}$  and  $|S_{far,T}| \leq k \cdot w(p_T^T)$ .

Proof. By Claims 1 and 2,

$$\text{cost}(S_{far,T}; p_T^T) = \sum_{i:p(y_i) \neq p_T^T} \text{cost}(S_{T_i}; p_T^T) \leq k \cdot (\beta_{T+1} + 2)\text{OPT}$$

By Claim 3,

$$|S_{far,T}| = \sum_{i:p(y_i) \neq p_T^T} |S_{T_i}| \leq k \cdot w(p_T^T). \quad \triangleleft$$

This concludes the proof of the claim, thus also of the lemma. ◀

## F Proof of Lemma 8

**Proof of Lemma 8.** As in Lemma 7, let  $C_j = C_j^T \setminus C_j^{T^-}$  and let  $S_{j_i}$  be the set of elements in  $C_j$  assigned to  $y_i$  in the clustering of  $X(T) \setminus X(T^-)$  induced by  $P_T$ . Let  $S_{far,j} = \bigcup_{i:p(y_i) \neq p_j^T} S_{j_i}$ ,  $S_{near,j} = \bigcup_{i:p(y_i) = p_j^T} S_{j_i}$ , and  $S_j$  be the elements in  $C_j$  that are assigned to  $p_j^T$  in the clustering of  $X(T) \setminus X(T^-)$  induced by  $P_T$ .

## 20:20 Online $k$ -Median with Consistent Clusters

The proof is by induction. We have that

$$|C_j^T| = |C_j^{T-}| + |C_j| = |C_j^{T-}| + |S_{far,j}| + |S_{near,j}| + |S_j| \quad (1)$$

(Note we use that there are no points in  $C_j$  that are assigned to  $p_{j'}^T$ ,  $j' \neq j$ , in the offline clustering induced by  $P_T$ , due to the greedy labelling rule. This is true as long as in the offline clustering induced by  $P_T$  we break ties consistent with how the online algorithm breaks ties.)

First, we bound the last three terms. Let  $w^t$  denote the natural weights at the end of phase  $t$ .

$$|S_{far,j}| \leq k \cdot w^T(p_j^T) \leq k \cdot w^T(c_j^T) \quad (2)$$

where the first inequality follows from Lemma 7 and the second inequality follows from the definition (‡) of estimated center. Next,

$$|S_{near,j}| = \sum_{i:p(y_i)=p_j^T} |S_{ji}| \leq \sum_{i:p(y_i)=p_j^T} w^T(y_i) \leq k \cdot w^T(c_j^T) \quad (3)$$

where the second inequality follows from the definition of  $w^T$ . The third inequality follows from the definitions of attachment digraph and estimated center: If  $y_i \in \delta^-(p_j^T)$ , then  $w^T(y_i) \leq w^T(p_j^T)$  by construction of the attachment digraph  $D(T)$ . Otherwise,  $y_i \in \delta^+(p_j^T)$ , so by (‡),  $w^T(c_j^T) \geq w^T(y_i)$ . Finally,

$$|S_j| \leq w^T(p_j^T) \leq w^T(c_j^T) \quad (4)$$

where the first inequality is by the definition of  $w^T$  and the second inequality from (‡).

For simplicity, let  $h(t, k) = (2k + 1)t$ . Now we need to bound  $|C_j^{T-}|$  in terms of  $w^T(c_j^T)$ . If  $j \notin [T^-]$ , then  $|C_j^{T-}| = 0$ . So assume  $j \in [T^-]$ . Inductively, we have that

$$|C_j^{T-}| \leq h(T^-, k) \cdot w^{T-}(c_j^{T-}).$$

We will prove that

$$|C_j^{T-}| \leq h(T^-, k) \cdot w^T(c_j^T). \quad (5)$$

There are two cases to consider.

► **Case 1.** (a) holds in Lemma 6.

$$|C_j^{T-}| \leq h(T^-, k) \cdot w^{T-}(c_j^{T-}) \leq h(T^-, k) \cdot w^T(p_j^T) \leq h(T^-, k) \cdot w^T(c_j^T).$$

► **Case 2.** (b) holds in Lemma 6.

This means that  $c_j^{T-}$  is  $\beta_{T+1}$ -attached to  $p_j^T$  w.r.t.  $w^T$ . If  $w^T(c_j^{T-}) \leq w^T(p_j^T)$ , then  $w^T(c_j^{T-}) \leq w^T(c_j^T)$ . Otherwise,  $w^T(c_j^{T-}) > w^T(p_j^T)$ , so  $c_j^{T-} \in \delta^+(p_j^T)$ . By (‡),  $w^T(c_j^{T-}) \leq w^T(c_j^T)$ . In both cases we have  $w^T(c_j^{T-}) \leq w^T(c_j^T)$ , so building from the inductive assumption,

$$|C_j^{T-}| \leq h(T^-, k) \cdot w^{T-}(c_j^{T-}) \leq h(T^-, k) \cdot w^T(c_j^{T-}) \leq h(T^-, k) \cdot w^T(c_j^T)$$

which concludes the case. Putting equations (1), (2), (3), (4), (5) together gives

$$|C_j^T| \leq (h(T^-, k) + 2k + 1) \cdot w^T(c_j^T) \leq h(T, k) \cdot w^T(c_j^T) = (2k + 1) \cdot T \cdot w^T(c_j^T)$$

as desired. ◀

## G Proof of Lemma 9

**Proof of Lemma 9.** The proof is by induction. Let  $C_j$ ,  $S_{ji}$ , and  $S_{far,j}$  be as in Lemma 7. Define  $S_{near,j} = \bigcup_{i:p(y_i)=p_j^T} S_{ji}$  and  $S_j$  to be the elements in  $C_j$  that are assigned to  $p_j^T$  in the clustering of  $X(T) \setminus X(T^-)$  induced by  $P_T$ . Let  $w^t$  denote the natural weights at the end of phase  $t$ . First we need the following key claim.

▷ **Claim 1.** For any  $x, y \in \delta^+(p_j^T) \cup \delta^-(p_j^T) \cup \{p_j^T\}$ ,  $x$  and  $y$  are  $2\beta_{T+1}$ -attached w.r.t.  $w^T$ .

*Proof of Claim 1.* If  $x$  or  $y$  is  $p_j^T$ , then the claim automatically holds by Proposition 13. There are two other cases. The first case is, WLOG,  $x \in \delta^-(p_j^T)$ . Regardless of whether  $y$  is in  $\delta^-(p_j^T)$  or  $\delta^+(p_j^T)$ , the claim holds by Propositions 13 and 11. The second case is that  $x, y \in \delta^+(p_j^T)$ . We prove the stronger statement that  $x$  and  $y$  are  $\beta_{T+1}$ -attached w.r.t.  $w^T$ . Suppose to the contrary that  $x$  and  $y$  are  $\beta_{T+1}$ -well-separated. We claim that this implies

$$\{p_1^T, \dots, p_T^T\} \cup \{x, y\} \setminus \{p_j^T\} \quad (6)$$

is  $\beta_{T+1}$ -well-separated w.r.t.  $w^T$ ; this would give a contradiction, since if an Exchange Operation were available, it would have been executed. Now suppose that (6) does not hold. Then WLOG  $p_{j'}^T$  and  $x$  are  $\beta_{T+1}$ -attached w.r.t.  $w^T$ , for some  $j' \neq j$ . Since  $x \in \delta^+(p_j^T)$  and since  $x$  and  $p_{j'}^T$  are  $\beta_{T+1}$ -attached w.r.t.  $w^T$ , by Proposition 11,  $p_j^T$  and  $p_{j'}^T$  are  $2\beta_{T+1}$ -attached w.r.t.  $w^T$ . This contradicts that  $p_j^T$  and  $p_{j'}^T$  are  $\beta_T$ -well-separated w.r.t.  $w^T$ , since  $2\beta_{T+1} < \beta_T$ . This concludes the proof of the case and the claim. ◁

To bound the cost contribution of  $C_j^{T-}$ , we case on which statement holds in Lemma 6.

▶ **Case 1.**  $c_j^{T-}$  is  $\beta_{T+1}$ -attached to  $p_j^T$  w.r.t.  $w^T$  (i.e., (b) holds in Lemma 6).

Since in Case 1,  $c_j^{T-}$  is  $\beta_{T+1}$ -attached to  $p_j^T$  w.r.t.  $w^T$ ,  $c_j^{T-} \in \delta^+(p_j^T) \cup \delta^-(p_j^T)$ . Also,  $c_j^T$  by definition is in  $\delta^+(p_j^T) \cup \{p_j^T\}$ . So by Claim 1,  $c_j^{T-}$  is  $2\beta_{T+1}$ -attached to  $c_j^T$  w.r.t.  $w^T$ . Using this, we bound  $\text{cost}(C_j^{T-}; c_j^T)$ :

$$\begin{aligned} \text{cost}(C_j^{T-}; c_j^T) &\leq \text{cost}(C_j^{T-}; c_j^{T-}) + |C_j^{T-}| \cdot d(c_j^{T-}, c_j^T) \\ &\leq g(T^-, k) \cdot \text{OPT} + |C_j^{T-}| \cdot d(c_j^{T-}, c_j^T) \\ &\leq g(T^-, k) \cdot \text{OPT} + (2k+1) \cdot T^- \cdot w^{T-}(c_j^{T-}) \cdot d(c_j^{T-}, c_j^T) \\ &\leq g(T^-, k) \cdot \text{OPT} + (2k+1) \cdot T^- \cdot w^T(c_j^{T-}) \cdot d(c_j^{T-}, c_j^T) \\ &\leq g(T^-, k) \cdot \text{OPT} + (2k+1) \cdot T^- \cdot 2\beta_{T+1} \cdot \text{OPT} \end{aligned} \quad (7)$$

where the third inequality is due to Lemma 8.

▶ **Case 2.** (b) does not hold in Lemma 6, so (a) holds, i.e.,  $w^{T-}(c_j^{T-}) \leq w^T(p_j^T)$  and  $w^{T-}(c_j^{T-}) \cdot d(c_j^{T-}, p_j^T) \leq \beta_{T-}(T - T^-) \cdot \text{OPT}$ .

We bound  $\text{cost}(C_j^{T-}; c_j^T)$ :

$$\begin{aligned} \text{cost}(C_j^{T-}; c_j^T) &\leq \text{cost}(C_j^{T-}; c_j^{T-}) + |C_j^{T-}| \cdot d(c_j^{T-}, c_j^T) \\ &\leq g(T^-, k) \cdot \text{OPT} + |C_j^{T-}| \cdot d(c_j^{T-}, p_j^T) + |C_j^{T-}| \cdot d(p_j^T, c_j^T) \end{aligned} \quad (8)$$

and now we use the assumptions of the case to continue bounding from (8):

$$\begin{aligned} |C_j^{T-}| \cdot d(c_j^{T-}, p_j^T) &\leq (2k+1) \cdot T^- \cdot w^{T-}(c_j^{T-}) \cdot d(c_j^{T-}, p_j^T) \\ &\leq (2k+1) \cdot T^- \cdot \beta_{T-}(T - T^-) \cdot \text{OPT} \end{aligned} \quad (9)$$

## 20:22 Online $k$ -Median with Consistent Clusters

where the first inequality is due to Lemma 8. Next,

$$\begin{aligned} |C_j^{T^-}| \cdot d(p_j^T, c_j^T) &\leq (2k+1)T^- \cdot w^{T^-}(c_j^{T^-}) \cdot d(p_j^T, c_j^T) \leq (2k+1)T^- \cdot w^T(p_j^T) \cdot d(p_j^T, c_j^T) \\ &\leq (2k+1)T^- \cdot \beta_{T+1} \cdot \text{OPT} \end{aligned} \quad (10)$$

where the first inequality is due to Lemma 8 and the last inequality is due to Proposition 14. So combining (8), (9), (10) gives

$$\text{cost}(C_j^{T^-}; c_j^T) \leq g(T^-, k) \cdot \text{OPT} + (2k+1) \cdot T^- \cdot (\beta_{T^-}(T - T^-) + \beta_{T+1}) \cdot \text{OPT}. \quad (11)$$

Now we have bounds (7) and (11) for  $\text{cost}(C_j^{T^-}; c_j^T)$ . Recall that  $C_j^T = C_j^{T^-} \cup S_{far,j} \cup S_{near,j} \cup S_j$ . The following bounds will hold regardless of whether we are in Case 1 or 2. We have

$$\text{cost}(S_j; c_j^T) \leq \text{cost}(S_j; p_j^T) + |S_j| \cdot d(p_j^T, c_j^T) \leq 2\text{OPT} + w^T(p_j^T) \cdot d(p_j^T, c_j^T) \leq (2 + \beta_{T+1})\text{OPT} \quad (12)$$

$$\begin{aligned} \text{cost}(S_{near,j}; c_j^T) &= \sum_{i:p(y_i)=p_j^T} \text{cost}(S_{ji}; c_j^T) \leq \sum_{i:p(y_i)=p_j^T} \sum_{p \in S_{ji}} d(p, c_j^T) \\ &\leq 2\text{OPT} + \sum_{i:p(y_i)=p_j^T} w^T(y_i) \cdot d(y_i, c_j^T) \leq (2k\beta_{T+1} + 2)\text{OPT} \end{aligned} \quad (13)$$




where we have used Claim 1 and that  $|S_{ji}| \leq w^T(y_i)$ . Finally, by Lemma 7,

$$\text{cost}(S_{far,j}; c_j^T) \leq \text{cost}(S_{far,j}; p_j^T) + |S_{far,j}| \cdot d(p_j^T, c_j^T) \leq k(2\beta_{T+1} + 2)\text{OPT} \quad (14)$$

Combining (12), (13), (14) with (7) or (11) gives the sought bound:

$$\text{cost}(C_j^T; c_j^T) \leq [g(T^-, k) + g(k)]\text{OPT} \leq g(T, k) \cdot \text{OPT}. \quad \blacktriangleleft$$

# The Telephone $k$ -Multicast Problem

Daniel Hathcock   

Carnegie Mellon University, United States

Guy Kortsarz 

Rutgers University, Camden, United States

R. Ravi  

Carnegie Mellon University, United States

---

## Abstract

We consider minimum time multicasting problems in directed and undirected graphs: given a root node and a subset of  $t$  terminal nodes, multicasting seeks to find the minimum number of rounds within which all terminals can be informed with a message originating at the root. In each round, the telephone model we study allows the information to move via a matching from the informed nodes to the uninformed nodes.

Since minimum time multicasting in digraphs is poorly understood compared to the undirected variant, we study an intermediate problem in undirected graphs that specifies a target  $k < t$ , and requires the only  $k$  of the terminals be informed in the minimum number of rounds. For this problem, we improve implications of prior results and obtain an  $\tilde{O}(t^{1/3})$  multiplicative approximation. For the directed version, we obtain an *additive*  $\tilde{O}(k^{1/2})$  approximation algorithm (with a poly-logarithmic multiplicative factor). Our algorithms are based on reductions to the related problems of finding  $k$ -trees of minimum poise (sum of maximum degree and diameter) and applying a combination of greedy network decomposition techniques and set covering under partition matroid constraints.

**2012 ACM Subject Classification** Theory of computation → Routing and network design problems

**Keywords and phrases** Network Design, Multicast, Steiner Poise

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.21

**Category** APPROX

**Funding** *Daniel Hathcock*: Supported by the NSF Graduate Research Fellowship grant DGE-2140739  
*R. Ravi*: This material is based upon work supported in part by the Air Force Office of Scientific Research under award number FA9550-23-1-0031 to RR.

## 1 Introduction

We study an information spreading problem that captures applications in distributed computing [16] and keeping distributed copies of databases synchronized [2]. A given graph models a synchronous network of processors that exchange information in rounds. There are several models describing how information may be exchanged between processors in the graph. In this work, we focus on the classic *Telephone Model* [8]: during a round, each vertex that knows the message can send the message to at most one of its neighbors.

In the **Minimum Time Telephone Multicast** (MTM) problem, we are given a network, modeled by a directed or undirected graph  $G(V, E)$ , a root vertex  $r$  that knows a message, and a set  $S$  of terminals. The message must be transmitted from  $r$  to  $S$  under the telephone model. In every round, there is a set of vertices  $K \subseteq V$  that know the message (initially  $K = \{r\}$ ), and the communication in a given round is described by a matching  $\{(k_1, v_1), \dots, (k_\ell, v_\ell)\}$  between some pairs of vertices  $k_i \in K$  and  $v_i \notin K$  for which  $k_i v_i \in E$ . In the directed setting, edge  $k_i v_i$  must be directed from  $k_i$  to  $v_i$ . Following this round, all of the matched vertices  $\{v_i\}$  are added to  $K$ . When  $S = V$  this problem is called **The Minimum Time Broadcast** (MTB) problem.



© Daniel Hathcock, Guy Kortsarz, and R. Ravi;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 21; pp. 21:1–21:16



Leibniz International Proceedings in Informatics

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

## 21:2 The Telephone $k$ -Multicast Problem

The best-known approximation ratio for the MTM problem on an undirected graph is  $O(\log t / \log \log t)$  [5], where  $t = |S|$ . In [4], it is shown that unless  $P = NP$ , the MTB problem admits no  $3 - \epsilon$  approximation for any constant  $\epsilon$ . For directed graphs, the Minimum Time Broadcast problem admits an  $O(\log n)$  approximation [4] in an  $n$ -node graph. The same paper shows that unless  $P = Quasi(P)$  the problem admits no better than  $\Omega(\sqrt{\log n})$  approximation.

However, for the directed case the multicast problem *seems* harder to approximate. The best-known approximation ratio for this problem is an additive  $O(\sqrt{t})$  guarantee (with poly-logarithmic multiplicative factor) [3]. This leaves a wide gap between the current best approximation algorithms for undirected versus directed multicast problems. In this work, we make progress toward closing that gap by studying an intermediate problem, the **Minimum Time Telephone  $k$ -Multicast** problem ( $k$ -MTM), defined below.

**Input:** A directed or undirected graph  $G(V, E)$  with root  $r$ , a collection of terminals  $S \subseteq V$  and a number  $k \leq |S|$ .  
**Required:** Send the message originating at  $r$  to *any*  $k$  terminals of  $S$  in the telephone model in a minimum number of rounds.

In terms of approximability, the undirected  $k$ -MTM problem lies between the undirected and directed MTM problems. Specifically, in [11] it is shown<sup>1</sup> that a  $\rho$ -approximation for directed MTM implies an  $O(\text{polylog } k)$ -approximation for undirected  $k$ -MTM, while it is immediate that any approximation for undirected  $k$ -MTM gives the same factor approximation for undirected MTM.

On the other hand, the directed version of the  $k$ -MTM problem generalizes all of the aforementioned problems.

**Applications.** Broadcast and multicast problems find numerous applications in distributed settings. For example, in the Network Aggregation problem, each user sends its data to a chosen central vertex  $r$ . This is equivalent to broadcasting in the local model for distributed computation (see [9]). Broadcasting is also crucial in Sensor Networks [13]. Another application is ensuring that the maximum information delay in vector clocks problems is minimized [12, 17].

One application of multicasting is to keep information across copies of replicated databases consistent, by broadcasting from the changed copy to the others [7, 14, 15]. If we are given a large set of  $t$  terminals of which we only want to keep replicated copies in some  $k$  of them, finding the best  $k$  to minimize the maximum synchronization time among these terminals corresponds to the  $k$ -MTM problem.

**Minimum Poise Trees.** Any telephone multicast schedule defines a tree rooted at  $r$ , spanning all terminals. The parent of a vertex  $u \neq r$  is defined to be the unique vertex that sends the message to  $u$ . Let  $T^*$  be the tree defined by the optimal schedule. The height of  $T^*$  (the largest distance in  $T^*$  from the root) is denoted by  $D^*$ . The largest out-degree<sup>2</sup> in  $T^*$  is denoted by  $B^*$ . The *poise* of  $T^*$  is defined as  $p^* = B^* + D^*$  [18]. Denote by OPT the number of rounds used by the optimal schedule. Since at every round, each informed vertex can send

---

<sup>1</sup> [11] deals with the degree-bounded versions of these problems, but their proof works as well for poise problems. See below for the connection between poise and  $k$ -MTM.

<sup>2</sup> For simplicity, we say degree instead of out-degree for the rest of the paper when discussing directed graphs.



the message to at most one neighbor,  $\text{OPT} \geq B^*$  and  $\text{OPT} \geq D^*$ . Hence, in general we have  $\text{OPT} \geq p^*/2$ . A partial converse is shown in [18]. A  $\rho$  approximation for the **Minimum Poise Steiner Tree** implies an  $O(\log t) \cdot \rho / \log \log t$  approximation for the MTM problem.

Following [18], approximating the  $k$ -MTM problem is equivalent (up to logarithmic factors in  $k$ ) to approximating the following **Minimum Poise Steiner  $k$ -Tree** problem:

**Input:** A directed or undirected graph  $G(V, E)$  with root  $r$ , a collection  $S \subseteq V$  of terminals, and a number  $k$ .  
**Required:** A  $k$ -tree rooted at  $r$ , namely a tree  $T'(V, E)$  containing paths from  $r$  to  $k$  of the terminals, with minimum poise.

In [11], they show that the approximability of minimum degree Steiner  $k$ -tree reduces to minimum degree group Steiner tree (which is a special case of minimum degree directed Steiner tree). Their reduction immediately extends to the minimum poise versions of these problems. Hence, the approximability of the undirected Minimum Poise Steiner  $k$ -Tree problem lies between the undirected Minimum Poise Steiner Tree problem and the directed Minimum Poise Steiner Tree problem (up to  $\log k$  factors). This implies the aforementioned analogous statement about the relationship between undirected  $k$ -MTM and the undirected/directed MTM problems.

We focus on approximating these poise problems.

► **Definition 1.1.** *A  $O(f(k))$ -additive approximation for the Minimum Poise Steiner  $k$ -Tree problem returns a tree  $T$  with  $k$  terminals, with maximum degree<sup>3</sup>  $\tilde{O}(B^*) + O(f(k))$  and height  $O(D^*)$ .*

## 1.1 Our results

We give an  $O(\sqrt{k})$ -additive approximation for the directed versions.

► **Theorem 1.2.** *Minimum Poise Steiner  $k$ -tree problem on directed graphs admits a polynomial time  $\tilde{O}(k^{1/2})$ -additive approximation. This implies the same approximation for the Minimum Time Telephone  $k$ -multicast problem.*

The second part of the statement follows from [18].

In [10], a multiplicative  $O(\sqrt{k})$ -approximation is given for the directed Min-Max Degree  $k$ -Tree problem, which asks to find a tree spanning  $k$  terminals while minimizing the maximum degree. Their algorithm iteratively finds trees containing  $\sqrt{k} \cdot B^*$  terminals, and uses flows to connect them to the root. Our directed result is more general than that of [10] in that it can handle both degree bounds and height bounds. Moreover, our approximation for degree is stronger, since we get an additive  $O(\sqrt{k})$  approximation. Therefore, it may be better than the approximation of [10] in the case that  $B^*$  is large. Our approximation ratio for the diameter is constant.

Our result is also more general than the  $O(\sqrt{t})$ -additive approximation for directed MTM of [3], as it handles the  $k$ -tree version of the problem, and recovers the same  $O(\sqrt{t})$ -additive approximation in the case  $k = t$  (up to logarithmic factors). In [3], the so-called *multiple set-cover* problem is used, a variant of set cover, while our result uses max coverage subject to a matroid constraint.

---

<sup>3</sup> The  $\tilde{O}$  notation hides poly-logarithmic factors in  $k$

For undirected graphs, we give an  $\tilde{O}(t^{1/3})$  approximation, which is a better ratio in the worst case if  $k$  is close to  $t$ . This represents progress toward closing the gap between the approximability of undirected and directed MTM, since in [11] it is shown that the undirected  $k$ -MTM problem lies between undirected and directed MTM in terms of approximability.

► **Theorem 1.3.** *The Minimum Poise Steiner  $k$ -tree problem on undirected graphs admits a polynomial time  $\tilde{O}(t^{1/3})$  approximation, and therefore the Minimum Time Telephone  $k$ -multicast problem admits the same approximation.*

The  $O(\sqrt{k})$  additive ratio can be as bad as  $\Omega(\sqrt{t})$  multiplicative ratio, if  $B^*$  is constant and  $k = \Omega(t)$ . Therefore, in the worst case, an  $\tilde{O}(t^{1/3})$  approximation is a better ratio. In addition, if  $B^* = o(t^{1/6})$  and  $k = \Omega(t)$ , the multiplicative ratio gives a better additive ratio.

## 1.2 Technical Overview

For the directed case, our techniques are based on [3]. However, our problem is harder since it is not clear which  $k$  terminals to choose. An important difference is that we use an approximation algorithm for maximizing set coverage (a submodular function) under matroid constraints [1]. The multiplicative approximation for the undirected case builds on this, and requires several graph decomposition techniques to be carefully combined.

For both results, we denote the maximum degree as  $B^*$  and height as  $D^*$  of an optimal minimum poise tree  $T^*$ . It can be assumed that  $D^*$  and  $B^*$  are known by trying all possibilities, as there are only polynomially many. Moreover, since  $D^*$  is known, all vertices of distance greater than  $D^*$  from the root may be removed.

**Directed Min-Poise Steiner  $k$ -Tree.** In order to get an  $O(\sqrt{k})$  additive approximation for the directed min-poise Steiner  $k$ -tree problem, we employ a greedy strategy. We iteratively find a collection of vertex-disjoint trees, each covering (i.e., containing) exactly  $\sqrt{k}$  terminals and of height at most  $D^*$ , until no more can be found. We call these *good trees*.

In the case that at least  $\sqrt{k}$  many good trees are found, an additive  $O(\sqrt{k})$ -approximation follows by taking any  $\sqrt{k}$  of the good trees along with shortest paths from the root  $r$  to the roots of each of these trees. This yields a subgraph (not necessarily a tree, since the shortest paths may not be disjoint from the good trees) with maximum out-degree at most  $2\sqrt{k}$ , and radius (maximum distance from  $r$ ) at most  $2 \cdot D^*$ . Moreover, the subgraph contains  $k$  terminals. Now the non-disjointness may be overcome by returning a shortest path tree spanning this subgraph. This gives the desired approximation.

In the other case that fewer than  $\sqrt{k}$  good trees are found, we may still connect them to the root via shortest paths. This gives a subgraph of low poise, but does not yet cover  $k$  terminals. If  $k_1 < k$  terminals are covered, we must determine how to cover  $k - k_1$  additional terminals without inducing high degree or height.

This is the main technical contribution of the directed result: we can recast the covering of  $k - k_1$  additional terminals as a set cover instance, and the desired poise guarantees can be obtained by imposing a partition matroid constraint on the sets in the instance. Then, an algorithm for approximating submodular function maximization subject to a matroid constraint [1] is applied. To the authors' knowledge, partition matroid constrained set coverage has not previously been used for multicasting problems.

**Partition Matroid Set Coverage Procedure.** Suppose we are given a partition of the graph into  $A \cup C = V$  with  $r \in A$ , such that all of  $A$  is reachable with low poise and contains  $k_1$  terminals. We want to cover at least  $k - k_1$  terminals in  $C$  with low poise, and we know that there exists a tree  $T^*$  rooted at  $r$  which does so.

Say that a node  $c \in C$  covers all the terminals in  $C$  that it can reach within distance  $D^*$ . In this way, we define a set cover instance over the ground set of terminals in  $C$  in which each set is identified by an edge  $(a, c)$  between a node  $a \in A$  and a node  $c \in C$ . The set corresponding to  $(a, c)$  contains all terminals covered by  $c$ . Defining the sets this way allows us to enforce degree constraints on the nodes in  $A$ , since the sets can be partitioned by their member in  $A$ . That is, we form a partition with the parts  $X(a) = \{(a, c) : c \in C, ac \in E\}$  for each  $a \in A$ . We now impose the constraint that at most  $B^*$  sets may be chosen from any part  $X(a)$ , reflecting the desired degree constraint. A *partition matroid* captures choosing at most a certain number of elements from each part of a partitioned set. Hence we have described a set cover instance with a partition matroid constraint and a coverage requirement of  $k - k_1$ .

The problem of selecting sets to maximize the number of terminals covered subject to the matroid constraint is a special case of submodular function maximization subject to a matroid constraint. Moreover,  $T^*$  provides a certificate that there exists a collection of sets satisfying the matroid constraint and covering at least  $k - k_1$  terminals in  $C$ . Hence, we may apply the  $(1 - \frac{1}{e})$ -approximation for this problem [1] (the simple greedy strategy giving a  $\frac{1}{2}$ -approximation [6] would also suffice here) to find a collection of sets satisfying the matroid constraint and covering at least  $(1 - \frac{1}{e}) \cdot (k - k_1)$  terminals in  $C$ .

Given the choice of sets  $(a, c)$  by the algorithm, we identify a set of edges that may be added to extend our subgraph to cover these terminals. These newly covered terminals are then removed, and the process repeated. In each round, we can cover a constant fraction of the desired number of terminals, so we need only  $O(\log k)$  rounds. Moreover, any given round induces additional degree of only  $B^*$  on nodes in  $A$ . The degree induced on nodes in  $C$  depends on the size of the parts  $X(a)$ , and this can be bounded in our applications (e.g., by  $\sqrt{k}$  in the directed setting described above, since the greedy strategy ensures that all nodes in  $c$  can reach at most  $\sqrt{k}$  terminals within distance  $D^*$ ). Finally, the distance from the root of any node added is  $O(D^*)$ , so in total the poise of the subgraph remains low. In the end, we again output a shortest path tree spanning this subgraph.

**Improvement in Undirected Graphs.** In the undirected setting, the result can be improved by taking advantage of the fact that if a good (low-poise) tree covering many terminals is found, then we need only cover *any* node in that tree in order to cover all of those terminals with low poise (as opposed to the directed case where we would have to cover the *root* of that tree). Essentially, we may contract the tree and treat the contracted node as containing many terminals.

Specifically, we will maintain a set  $R$  of nodes that we have covered so far with low poise (by contracting, we can think of this simply as the root  $r$ ). We first group the terminals in the remaining graph  $C = V \setminus R$  as before by greedily finding disjoint trees of low poise, now each containing  $t^{1/3}$  terminals, called *small trees*. Note that some terminals may not lie in any small tree. If the algorithm finds fewer than  $t^{1/3}$  small trees, then the same matroid-constrained covering procedure from above can be applied to immediately get an *additive*  $O(t^{1/3})$ -approximation.

On the other hand, if there are many small trees, we show that progress can be made by either covering or discarding a large number of terminals at once. If we are able to aggregate  $t^{1/3}$  small trees into a single tree within a distance  $D^*$ , we have covered  $t^{2/3}$  terminals and hence made sufficient progress in coverage: we can repeat this at most  $t^{1/3}$  times to finish, inducing at most  $t^{1/3}$  degree at the root to reach these trees. However, we may have the additional complexity of the optimal tree containing terminals that are not in one of

these small trees we computed in  $C$ . We handle this case by using the matroid-constrained coverage procedure to extract as many terminals as any optimal solution might cover from the small trees while staying within the degree and height bounds, and then discarding all the terminals from *all* of the unused small trees. Since the number of small trees (each with  $t^{1/3}$  terminals) is  $\Omega(t^{1/3})$ , this allows us to bound the number of such discarding iterations by  $O(t^{1/3})$ . In summary, we employ  $O(t^{1/3})$  iterations of either covering or discarding  $t^{2/3}$  terminals in the algorithm leading to the claimed  $O(t^{1/3})$  multiplicative guarantee. Over the course of these iterations, the total degree accumulated by any node will be at most  $\tilde{O}(t^{1/3}) \cdot B^*$  (Note this guarantee is now multiplicative, since a node can gain  $\tilde{O}(B^*)$  degree in each of the  $t^{1/3}$  covering iterations).

Finally, we remark that the improved guarantee in this setting is in terms of  $t$ , the total number of terminals, rather than  $k$ . This is because our algorithm relies on removing a large number of terminals from the entire set of  $t$  terminals, without necessarily covering all of them.

## 2 Preliminaries

Let  $\text{dist}(u, v)$  denote the number of edges in the shortest path from  $u$  to  $v$  in  $G$ . We denote by  $G[U]$  the graph induced by  $U$ , and by  $\text{dist}_{G[U]}(u, v)$  the distance from  $u$  to  $v$  in the graph  $G[U]$ . Recall that we denote the minimum poise tree by  $T^*$ , its maximum degree by  $B^*$ , and its height by  $D^*$ .

► **Assumption 2.1.** *Removing vertices of distance more than  $D^*$  from the root  $r$  in  $G$  does not change the optimal solution. Hence, we will assume for the rest of the paper that  $G$  only contains vertices of distance at most  $D^*$  from  $r$ .*

► **Remark 2.2.** For the rest of the paper, we assume that quantities such as  $\sqrt{k}$  are integral. Making the algorithm precise requires using  $\lceil \sqrt{k} \rceil$ . However, the changes are minimal and elementary.

For simplicity, we assume that every terminal has in-degree 1 and out-degree 0, by attaching new terminal vertices to every terminal (this only increases the poise by at most an additive constant). For undirected graphs, we assume that terminals have degree 1. Therefore, removing terminals can't turn a connected graph into a disconnected graph.

The input for the Set Cover problem is a universe  $\mathcal{U}$  and a collection  $\mathcal{S}$  of sets  $S_i \subseteq \mathcal{U}$ . We say that a set  $S_i$  covers all the elements that belong to this set. The goal is to find a sub-collection of sets  $\mathcal{S}' \subseteq \mathcal{S}$  of minimum size that covers all elements, namely,  $\bigcup_{S_i \in \mathcal{S}'} S_i = \mathcal{U}$ . The Set Coverage problem under matroid constraints has the input of Set Cover, and in addition, a matroid  $\mathcal{M}$  defined over the sets  $\mathcal{S}$ . The goal is to select an independent set  $\mathcal{I}$  in the Matroid so that  $|\bigcup_{S_i \in \mathcal{I}} S_i|$  is maximum. A partition matroid instance divides  $\mathcal{S}$  into pairwise disjoint collections of sets  $\mathcal{S}_i$ , whose union is all of  $\mathcal{S}$ . For every collection  $\mathcal{S}_i$ , there is a bound  $p_i$  on the number of sets that can be selected from  $\mathcal{S}_i$ . A collection of sets containing at most  $p_i$  sets from each  $\mathcal{S}_i$  is precisely an independent set in the partition matroid. The goal is to find an independent set in the partition matroid that covers the largest number of elements. This problem is a special case of maximizing a submodular function under matroid constraints. The greedy algorithm achieves a  $1/2$ -approximation for this problem [6], and is sufficient for our purposes. It is also known that the problem admits a polynomial time  $1 - 1/e$ -approximation [1], which may be used for improved constants. The procedure of [1] is one of the main tools in our algorithm. We called this procedure the **Matroid procedure**.

### 3 The Partition Matroid Cover Algorithm

In the next two sections, our algorithms for both the directed and undirected cases define a disjoint partition of the graph vertices into  $A \cup C = V$ . The root  $r$  always belongs to  $A$ , and we will ensure that all of  $A$  can be covered by a low poise tree rooted at  $r$ . In this section, we discuss how to cover sufficiently many terminals from  $C$  with low poise by connecting them to the root through  $A$ . We do this by defining an instance of the Set Coverage problem under a partition matroid constraint<sup>4</sup>.

► **Definition 3.1.** *Define a Set Coverage instance as follows.*

- *The items are  $S \cap C$  (the terminals in  $C$ ).*
- *The sets (also called pairs) are  $\mathcal{S} = \{(a, c) \mid a \in A, c \in C, \text{ and } ac \in E\}$  where  $(a, c)$  covers a terminal  $t \in S \cap C$  if  $\text{dist}_{G[C]}(c, t) \leq D^*$ .*

The partition matroid is defined as follows.

► **Definition 3.2.**  *$\mathcal{S}$  is partitioned into collections*

$$X(a) = \{(a, c) \mid c \in C \text{ and } ac \in E\}$$

for every  $a \in A$ . The bound on the number of sets to be chosen from  $X(a)$  is  $B^*$ .

By definition, the partition is disjoint and therefore, we have a valid partition matroid. Recall that  $r \in A$ . See Algorithm 1 for a description of the Procedure **PMCover**.

■ **Algorithm 1** **PMCover**.

---

**input** : Graph  $G(V, E)$  with terminals  $S$  and  $V$  partitioned into  $A \cup C$ , and a number  $k$ .

**output** : A collection of pairs of the form  $(a, c)$  with  $a \in A$  and  $c \in C$ .

- 1  $\mathcal{E}' \leftarrow \emptyset, S' \leftarrow S \cap C$ .
- 2 **while**  $k > 0$  **do**
- 3     Define the partition matroid Set Coverage instance from  $A, C, S'$  as above with sets  $\mathcal{S}'$  and apply Procedure **Matroid** of [1] to find an independent set of approximately maximum coverage. Let  $\mathcal{I}$  be the independent set it returns.
- 4      $\mathcal{E}' \leftarrow \mathcal{E}' \cup \mathcal{I}$ .
- 5     Decrease  $k$  by the number of terminals covered by  $\mathcal{I}$ .
- 6     Remove the terminals covered by  $\mathcal{I}$  from  $S'$ .
- 7 **return**  $\mathcal{E}'$ .

---

### Analysis

We will show that for every  $a \in A$ ,  $|X(a) \cap \mathcal{E}'| \leq O(\log k) \cdot B^*$ . This will be used to argue that if  $(a, c) \in \mathcal{E}'$ , we later may make  $a$  the parent of  $c$  in the tree we build without incurring high degree.

<sup>4</sup> Note that the parameter  $k$  represents the *remaining* number of terminals we need to cover. Given a partition  $A, C$  we will assume that all terminals in  $A$  have been spanned, and thus we need to cover  $k$  terminals in  $C$ . That is, if  $A$  has  $k_1$  terminals for some  $k_1 < k$ , we will set  $k \leftarrow k - k_1$ . Note that we are guaranteed that  $C \cap T^*$  contains at least  $k - k_1$  terminals supplying a feasible solution.

► **Definition 3.3.** Define a mapping from terminals in  $T^* \cap C$  to  $\mathcal{S}'$  as follows. For a terminal  $t$ , let  $a = a_t$  be the vertex  $a \in A$  that is an ancestor of  $t$  in  $T^*$  and among them  $\text{dist}_{T^*}(a, t)$  is minimum. This vertex is well defined since  $r \in A$  is the root of  $T^*$ . Let  $c = c_t$  be the child of  $a$  in  $T^*$  that is an ancestor of  $t$ . Define  $f(t) = (a, c)$ .

▷ **Claim 3.4.** There exists an independent set  $\mathcal{I}^*$  in the partition matroid that covers at least  $k$  terminals in  $C \cap S$ .

*Proof.* We show that every terminal in  $t \in T^* \cap C$  is covered by some set. Let  $a = a_t$  and let  $c = c_t$ . Since  $a$  has minimum distance to  $t$  from all vertices in  $A$ , the path from  $c$  to  $t$  belongs to  $G[C]$ . The number of edges in the path between  $c$  and  $t$  is at most  $D^* - 1$ . This implies that the set  $(a, c)$  covers  $t$ . Create a set  $\mathcal{I}^* = \{f(t) \mid t \in T^* \cap S \cap C\}$ . We note that  $f(t) = f(t') = (a, c)$  may hold for two different terminals, but  $\mathcal{I}^*$  includes every such pair  $(a, c)$  once (namely,  $\mathcal{I}^*$  is a set and not a multiset). For any  $a \in A$ , the number of different pairs of the form  $(a, c_1), (a, c_2), \dots$  in  $\mathcal{I}^*$  can't be more than  $B^*$ , because every such pair increases  $a$ 's degree in  $T^*$  by 1. Thus,  $\mathcal{I}^*$  is independent in the partition matroid. Since all terminals in  $T^* \cap C$  are covered,  $k$  terminals are covered. ◁

▷ **Claim 3.5.** Procedure `PMCover` returns a collection of pairs  $\mathcal{E}'$  so that for every  $a \in A$ ,  $X(a) \cap \mathcal{E}' = O(\log k) \cdot B^*$  and  $\mathcal{E}'$  covers  $k$  terminals. Thus if in some tree, vertex  $a \in A$  is made the parent of all  $c$  for which  $(a, c) \in \mathcal{E}'$ , the degree of  $a$  will be bounded by  $O(\log k) \cdot B^*$ .

*Proof.* Since Procedure `Matroid` returns an independent set in the partition matroid, at every iteration we have  $|X(a) \cap \mathcal{I}| \leq B^*$ . Claim 3.4 and the guarantee of Procedure `Matroid` by [1] imply that  $(1 - 1/e)k$  terminals are covered. Let  $k_{\text{or}} \leq k$  be the original number of terminals to be covered and  $k_{\text{new}}$  the number of terminals to be covered in a given iteration. Then in the next iteration,

$$k_{\text{new}} \leftarrow k_{\text{new}} - (1 - 1/e)k_{\text{new}} = \frac{k_{\text{new}}}{e}.$$

Therefore, after  $i$  iterations,  $k_{\text{or}}/e^i$  terminals remain to be covered. Hence, the number of iterations is  $O(\log k)$ . The claim follows. ◁

## 4 Approximating the poise for directed graphs

Our algorithm maintains a set  $A$  (initialized with the root  $r$ ) containing the terminals covered with low poise so far, and  $C = V \setminus A$ . Consider a set  $C$  and the graph  $G[C]$  induced by  $C$ .

► **Definition 4.1.** Denote by  $T(c)$  the coverage tree of  $c$  in  $G[C]$  formed by taking a shortest path from  $c$  to every terminal within distance  $D^*$ . A vertex  $c \in C$  is called  $\rho$ -good (with respect to  $C$ ) if there are at least  $\rho$  terminals in  $T(c)$ . A  $\rho$ -good tree is a tree rooted at some  $c$  (not necessarily  $T(c)$ ) with exactly  $\rho$  terminals and height at most  $D^*$ .

By assumption, the out-degree of terminals is 0. Therefore all terminals are leaves. Since we may discard non-terminal leaves, a  $\rho$ -good trees contains exactly  $\sqrt{k}$  leaf terminals.

► **Definition 4.2.** A set  $C$  of vertices, is a  $\rho$ -packing if there is no  $\rho$ -good vertex in  $C$ .

► **Definition 4.3.** Let  $\{T_i\}$  be a collection of vertex disjoint trees and let  $A$  be the set of vertices in  $\bigcup_i T_i$ . Let  $C = V - A$ . Then  $A, C$  is a  $\rho$ -additive partition if:

1. The trees  $T_i$  are  $\rho$ -good with respect to  $V$ , and are all vertex-disjoint.
2. There are at most  $\rho$  trees  $T_i$ .
3.  $C$  is a  $\rho$ -packing.

Let  $q_i$  be the root of  $T_i$ . Intuitively, since there are at most  $\rho$  trees  $T_i$ , we can add a shortest path  $P_i$  from the root  $r$  to each  $q_i$ , giving a tree rooted at  $r$  with low poise covering terminals in  $A$ . In addition, since  $C$  is a  $\rho$ -packing, at least  $k$  (meaning the number of *remaining* terminals to cover after covering those in  $A$ ) of  $C$ 's terminals can be covered with some collection of low poise trees. In particular, since every  $c \in C$  is not  $\rho$ -good, all of the coverage trees  $T(c)$  have max degree at most  $\rho$ .

The algorithm attempts to find a  $\rho$ -additive partition. It greedily finds  $\rho$ -good trees, and removes them until the set  $C$  that remains is a  $\rho$ -packing. Then the procedure **PMCover** can be used to connect the low poise trees covering  $A$  and  $C$ . However, there may be too many  $\rho$ -good trees in  $A$  for  $(A, C)$  to be a  $\rho$ -additive partition. In this case, it simply connects the root to any  $\rho$  of the trees  $T_i$ . By choosing  $\rho = \sqrt{k}$ , this ensures enough terminals are covered. See Algorithm 2 for a precise description of the Procedure **Directed**.

■ **Algorithm 2** **Directed**.

---

```

input : Graph  $G(V, E)$  with terminals  $S$ , and a number  $k$ .
output: A Steiner  $k$ -tree of  $G$ .

1 Set  $\rho = \sqrt{k}$ .
  /* Greedy Packing */
2 Let  $A = \{r\}$ , and  $C = V - \{r\}$ .
3 while  $C$  is not a  $\rho$ -packing do
4   Find a  $\rho$ -good tree  $T$  in  $G[C]$ .
5   Remove the vertices of  $T$  from  $C$  and add them to  $A$ .
6 Let  $\{T_i\}$  denote the set of  $\rho$ -good trees found.
  /* Many Trees */
7 if the number of  $\rho$ -good trees found is at least  $\rho$ , then
8   Choose any  $\rho$  of the trees  $\{T_i\}$  in  $A$ , and form the subgraph  $H \subseteq G$  by including
   the root  $r$ , the chosen trees, and a shortest path from  $r$  to the root  $q_i$  of each
   chosen tree  $T_i$ .
9   return a shortest path tree of  $H$  rooted at  $r$ .
  /* Few Trees */
10 else
11   The number of  $\rho$ -good trees found is at most  $\rho$ , so  $(A, C)$  is a  $\rho$ -additive partition.
   Apply the Procedure Complete on  $(A, C)$ , and return the resulting tree.

```

---

In the case that a  $\rho$ -additive partition  $(A, C)$  is found, we use the Procedure **Complete**, described in Algorithm 3. See Figure 1 for a depiction of the algorithm at this step.

## Analysis

For a directed tree,  $T$ , let  $\deg_T(v)$  be the (out-)degree of the vertex in  $T$ . Now say that we run step *Greedy Packing* of **Directed** with  $\rho = \sqrt{k}$ .

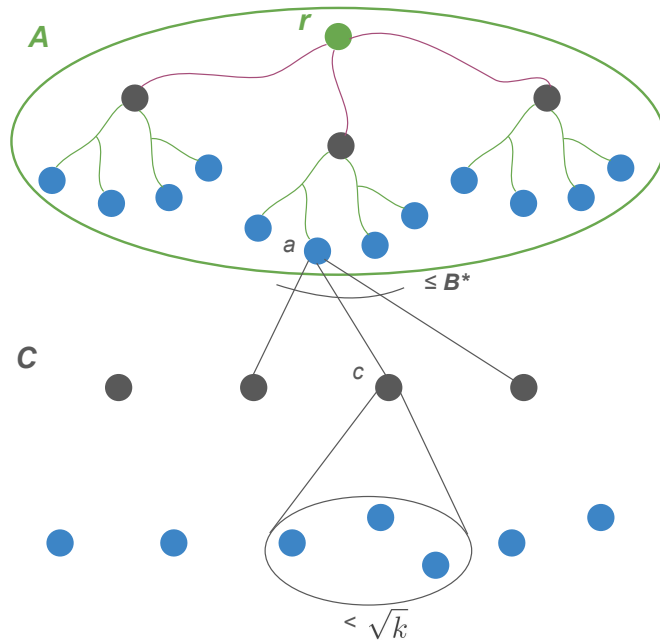
▷ **Claim 4.4.** If Procedure **Directed** finds at least  $\rho$   $\rho$ -good trees, then step *Many Trees* of Procedure **Directed** returns a tree with at least  $k$  terminals, maximum degree  $O(\sqrt{k})$ , and height  $O(D^*)$

*Proof.* Since each tree  $T_i$  is  $\sqrt{k}$ -good, it contains  $\sqrt{k}$  terminals. Hence, the graph  $H$  contains at least  $k$  terminals, each of which can be reached by a path from the root. So the returned shortest path tree of  $H$  has at least  $k$  terminals, as desired

## 21:10 The Telephone $k$ -Multicast Problem

■ **Algorithm 3** Complete.

- 
- input** : Graph  $G(V, E)$  with terminals  $S$ , a  $\rho$ -additive partition  $(A, C)$ , and a number  $k$ .
- output** : A Steiner  $k$ -tree of  $G$ .
- 1 Apply the procedure **PMCover** with partition  $(A, C)$  to get  $\mathcal{E}'$ .
  - 2 Let  $\mathcal{E} = \{ac : (a, c) \in \mathcal{E}'\}$ , the set of arcs corresponding to sets in  $\mathcal{E}'$ .
  - 3 Form the graph  $H_C$  on vertex set  $C \cup \{r'\}$ , where  $r'$  is a new node. For each  $c \in C$  appearing in some  $ac \in \mathcal{E}$ , include in  $H_C$  the arc  $(r', c)$  and the coverage tree  $T(c)$ . Take a shortest path tree on  $H_C$  rooted at  $r'$ , and let  $T_C$  be all of the edges from this tree in  $G[C]$ .
  - 4 Form the subgraph  $H \subseteq G$  by including the root  $r$ , each  $\rho$ -good tree  $T_i$  from  $A$  and a shortest path from  $r$  to its root  $q_i$ , the edges from  $\mathcal{E}$ , and the edges from  $T_C$ .
  - 5 **return** a shortest path tree of  $H$  rooted at  $r$ .
- 



■ **Figure 1** A depiction of the algorithm in the case that a  $\rho$ -additive partition is found. The set  $A$  includes the root  $r$  and all  $\sqrt{k}$ -good trees found, while  $C$  contains the remaining vertices. Terminals are depicted in blue. Short paths from  $r$  to the roots of the good trees are added (in red). Since  $C$  is a  $\sqrt{k}$ -packing, each vertex  $c \in C$  can reach less than  $\sqrt{k}$  terminals within distance  $D^*$ . Hence, we can run the **PMCover** procedure, with each iteration enforcing a degree constraint of  $B^*$  on each node in  $A$ , as shown.

To bound the degrees in the returned tree, we just bound the degrees in  $H$ . The good trees  $T_i$  are disjoint, and each have maximum degree at most  $\sqrt{k}$ . Moreover, there are  $\sqrt{k}$  of them, so there are only  $\sqrt{k}$  shortest paths to their roots. Therefore, the degree contributed to any node  $v \in H$  is at most  $\sqrt{k}$  from the  $T_i$ , and at most 1 for each shortest path, for a total of  $\deg_H(v) \leq 2\sqrt{k}$ . Finally, each tree  $T_i$  in  $H$  has height at most  $D^*$ , while each shortest path from the root to some  $q_i$  has length at most  $D^*$  (by Assumption 2.1), so the returned shortest path tree has height at most  $2 \cdot D^*$ .  $\triangleleft$



▷ **Claim 4.5.** If Procedure **Directed** finds less than  $\rho$   $\rho$ -good trees, then Procedure **Complete** finds a tree rooted at  $r$  with maximum degree  $O(\log k) \cdot B^* + O(\sqrt{k})$ , and height  $O(D^*)$  that contains at least  $k$  terminals of  $C \cap S$ .

*Proof.* First, observe that  $H$  contains all terminals in  $A$ , as well as those terminals in  $C$  covered by procedure **PMCover**. In particular, by Claim 3.5,  $H$  contains at least  $k$  terminals, so the returned shortest path tree does as well.

Now we bound the degrees of nodes in the returned tree. The  $T_i$  making up  $A$  are disjoint  $\sqrt{k}$ -good trees each having maximum degree at most  $\sqrt{k}$ . And there are less than  $\sqrt{k}$  of them, so we add at most  $\sqrt{k}$  shortest paths to their roots  $q_i$ . Hence, for each node  $v \in A$ , the contribution to the degree  $\deg_H(v)$  is at most  $\sqrt{k}$  from the  $T_i$ , at most 1 for each shortest path, plus the contribution from  $\mathcal{E}$ . By Claim 3.5, the edges of  $\mathcal{E}$  increase the degree of vertices in  $A$  by  $O(\log k) \cdot B^*$ , so in total  $\deg_H(v) \leq O(\log k) \cdot B^* + 2\sqrt{k}$  for each  $v \in A$ .

All other vertices in  $H$  lie in  $C$ , and so their degree comes only from the  $\sqrt{k}$  shortest paths (contributing at most 1 each), and the edges from  $T_C$ . Every coverage tree  $T(c)$  has depth at most  $D^*$  by definition. In particular, for any vertex  $c \in C$ , we must have  $\deg_{T_C}(c) \leq \sqrt{k}$ , since otherwise the subtree of  $T_C$  rooted at  $c$  has more than  $\sqrt{k}$  leaves, which can all be assumed to be terminals. But this means that  $c$  has more than  $\rho = \sqrt{k}$  terminals in  $C$  of distance at most  $D^*$ , contradicting that  $c$  is not  $\rho$ -good. Hence,  $\deg_H(c) \leq 2\sqrt{k}$  for every  $c \in C$ .

Finally, the height of the output tree is at most  $3 \cdot D^* + 1$ , because we get height  $D^*$ , from the trees  $T_i$ , height  $D^*$  from the shortest paths, height  $D^*$  from  $T_C$ , and an additional edge from  $\mathcal{E}$ . ◁

Therefore, in either case we return a tree with at least  $k$  terminals with maximum degree  $O(\log k) \cdot B^* + O(\sqrt{k})$  and height  $O(D^*)$ . This implies Theorem 1.2.

The following corollary is useful as it applies in case that the *Greedy Packing* step of Procedure **Directed** finds a  $\rho$ -additive partition (i.e., step *Few Trees* is executed) with some  $\rho$  that may be smaller than  $\sqrt{k}$ .

▶ **Corollary 4.6.** *If Procedure **Directed** finds a  $\rho$ -additive partition  $A, C$ , then there exists polynomial time  $\rho$ -additive approximation for the corresponding min poise  $k$ -tree problem.*

## 5 The undirected case

In this section, we provide our  $\tilde{O}(t^{1/3})$ -approximation algorithm for the Minimum Time Telephone  $k$ -multicast problem on undirected graphs with  $t$  terminals, proving Theorem 1.3.

**Preliminaries.** We assume (for convenience) that the root  $r$  is a non-leaf node in  $T^*$ . Recall that we assume that all terminals have degree 1. We can now assume that after rooting  $T^*$  at  $r$ , the set of leaves in  $T^*$  and the set of terminals in  $T^*$  is the same set. Also recall that the height of the tree  $T^*$  rooted at  $r$  is at most  $D^*$ , since the diameter of  $T^*$  is at most  $D^*$ .

**Algorithm outline.** The idea in the undirected case is that if a low-poise tree covering many terminals is found, then we need only cover *any* node in that tree in order to cover all of those terminals with low poise (as opposed to the directed case where we would have to cover the *root* of that tree). Essentially, we may contract the tree and treat the contracted node as containing many terminals.

Specifically, we will maintain a set  $R$  of nodes we have covered with low poise (by contracting, we can think of this simply as the root  $r$ ). We first partition the remaining graph  $C = V \setminus R$  as before by greedily finding small trees.

## 21:12 The Telephone $k$ -Multicast Problem

► **Definition 5.1.** We say that a tree is small size if it contains exactly  $t^{1/3}$  terminals. We say that a tree is large if it contains exactly  $t^{2/3}$  terminals

If this procedure succeeds in finding a  $t^{1/3}$ -additive partition, then we are done by Corollary 4.6. On the other hand, if we fail, we contract these small trees and show how to cover a sufficiently large number of them by either finding a single large tree reaching  $t^{1/3}$  of these small trees, or by applying the procedure **PMCover**. In either case, we may then remove *all* of the terminals from these small trees, contract the newly covered nodes into  $R$ , and iterate the entire process to cover the remaining terminals. In each iteration, we show the total number of terminals discarded is large, so there cannot be too many iterations, and hence not too much additional degree is incurred.

We first give a simple algorithm, Procedure **Small**, that finds trees  $\{T_i\}$  each with exactly  $t^{1/3}$  terminals (leaves).

■ **Algorithm 4** **Small**.

---

**input** : Graph  $G(V, E)$  with  $t$  terminals  $S$ , and a number  $k$ .

**output** : A collection of subtrees  $\{T_i\}$ , each with exactly  $t^{1/3}$  terminals, or a Steiner  $k$ -tree.

- 1 Apply step *Greedy Packing* from Procedure **Directed** on  $G$  with  $\rho = t^{1/3}$ . Denote the resulting trees as  $\{T_i\}$ .
  - 2 If the procedure succeeds in finding a  $t^{1/3}$ -additive partition, apply Procedure **Complete** on  $A, C$ , and **return** the resulting tree.
  - 3 Else, **return**  $\{T_i\}$
- 

In case that step *Greedy Packing* from Procedure **Directed** finds a  $t^{1/3}$ -additive partition  $A, C$ , we are guaranteed a  $t^{1/3}$ -additive ratio from Corollary 4.6. Hence, from now on we assume that step *Greedy Packing* from Procedure **Directed** gives more than  $t^{1/3}$  small trees  $T_i$ .

We will proceed to contract each of these small trees into super-terminals. The trees  $T_i$  that we compute, are built by step *Greedy Packing* from Procedure **Directed** with  $\rho = t^{1/3}$ . Hence, they have exactly  $t^{1/3}$  terminals/leaves. We contract the terminals of every  $T_i$  into a single super-terminal  $q_i$ . Denote by  $S(T_i)$  the terminals contained in  $T_i$  (i.e., those corresponding to  $q_i$ ). As mentioned in the outline, we have the possibility that the terminals of an optimal tree may only intersect with a few of these super-terminals. We capture this in the following definitions.

► **Definition 5.2.** We say that  $q_i$  is a true terminal if  $S(T_i) \cap T^* \neq \emptyset$ .

► **Definition 5.3.** Denote by  $k'$  the number of terminals in  $(\bigcup_i T_i) \cap T^*$ . Let  $\mu = \lceil k'/t^{1/3} \rceil$ .

From the definitions, we can see that  $T^*$  overlaps with at least  $\mu$  true terminals.

In the graph where the small trees have been contracted to super-terminals, we will attempt to find a  $t^{1/3}$ -packing of these super-terminals. For this, we generalize the definition of a  $t^{1/3}$ -packing in the set  $C$  with respect to the super-terminals.

► **Definition 5.4.** We say that  $c \in C$  is a  $t^{1/3}$ -good vertex with respect to the super-terminals  $\{q_i\}$  if there are at least  $t^{1/3}$  terminals  $q_i$  of distance at most  $D^*$  from  $c$ , in  $G[C]$ . If there are no  $t^{1/3}$ -good vertices in  $C$ ,  $C$  is called a  $t^{1/3}$ -packing with respect to  $\{q_i\}$ . If  $C$  is a  $t^{1/3}$ -packing, then  $R, C$  is called a  $t^{1/3}$ -additive partition with respect to  $\{q_i\}$ .

We can now describe the details of the rest of the undirected algorithm. Specifically, if Procedure **Small** fails to find a  $t^{1/3}$ -additive partition, then there are two possibilities. Either  $C = V \setminus R$  is a  $t^{1/3}$  packing with respect to the  $q_i$ , or otherwise there is a  $t^{1/3}$ -good vertex in  $C$ .

If  $C$  is a  $t^{1/3}$ -packing we apply Procedure **PMCover** on  $R, C$  with terminals  $\{q_i\}$  since  $R, C$  is a  $t^{1/3}$ -additive partition. The goal is covering  $\mu$  super-terminals. We know that  $T^*$  covers at least  $\mu$  true terminals  $q_i$ , so these can be reached with height  $D^*$  and maximum degree  $B^*$ . Therefore, our Procedure **PMCover** covers at least  $\mu$  super-terminals. Note that the number of original terminals we actually cover is  $\mu \cdot t^{1/3} \geq k'$ . This follows because each  $q_i$  represents a tree  $T_i$  that contains  $t^{1/3}$  terminals. We now discard all the terminals of  $\bigcup_i T_i$ . Since the number of  $T_i$  is at least  $t^{1/3}$ , the total number of discarded terminals is  $t^{2/3}$ .

The other case is that  $C$  is not a  $t^{1/3}$ -packing with respect to  $\{q_i\}$ . Let  $v \in C$  be a  $t^{1/3}$ -good vertex and let  $Q_v$  be the corresponding tree. Note that  $Q_v$  is a large tree since it spans  $t^{1/3}$  of the  $q_i$ , each representing  $t^{1/3}$  terminals. We connect  $r$  to  $Q_v$  via a shortest path  $P$  from  $r$  to  $Q_v$ , and contract  $r \cup P \cup Q_v$  into  $r$ . Then we discard the terminals of  $Q_v$ . Since  $Q_v$  is a large tree, the number of terminals discarded is  $t^{2/3}$ .

In summary, in both cases  $t^{2/3}$  terminals are discarded. Therefore the number of iterations in our algorithm is at most  $t^{1/3}$ .

The degree of vertices in  $R$  increases by  $O(\log k) \cdot B^*$  every time **PMCover** is applied. Alternatively, a large tree  $Q_v$  is created and we only need a path  $P$  from  $r$  to  $Q_v$ . This increases the degree of some vertices in  $R$  by exactly 2. This gives a total degree of  $2 \cdot t^{1/3}$  because of the bound on the number of iterations.

## The main procedure

Here we describe the precise algorithm for the undirected problem, Procedure **Undirected** in Algorithm 5.

### Analysis

▷ **Claim 5.5.**  $T^*$  contains at least  $\mu$  true terminals.

Proof. If the number of true terminals is at most  $\mu - 1$ , the number of terminals in  $\bigcup_i T_i$  is at most  $(\mu - 1) \cdot t^{1/3} < k'$  and this is a contradiction. ◁

▷ **Claim 5.6.** The number of iterations in Procedure **Undirected** is at most  $t^{1/3}$ .

Proof. If a tree  $Q_v$  is found, then it is a large tree hence it contains at least  $t^{2/3}$  terminals. These terminals are discarded in the iteration. Else, the terminals of  $S \cap \bigcup_i T_i$  are discarded and this, again, this removes  $t^{2/3}$  terminals, since we have at least  $t^{1/3}$  different small  $T_i$ 's (because procedure **Small** failed). Since in either case  $t^{2/3}$  terminals are discarded and the total number of terminals is  $t$ , the number of iterations is at most  $t^{1/3}$ . ◁

▷ **Claim 5.7.** Let  $v$  be a vertex so that  $v \notin R$ . A single iteration of Procedure **Undirected** increases  $v$ 's degree by at most  $2 \cdot t^{1/3} + 2$ . Moreover, if  $v$ 's degree increases,  $v$  is contracted into  $r$  in that iteration.

Proof. The degree of a vertex increases only if it belongs to a large tree  $Q_v$  (or its path  $P$  from  $r$ ), or it belongs to the subgraph  $Q$  computed by Procedure **PMCover**. In the first case, the degree increases by at most  $t^{1/3}$  from any one of the  $T_i$ 's in  $Q_v$ , at most  $t^{1/3}$  more for the paths from  $v$  to these  $T_i$ 's, and at most 2 more for the path  $P$  from  $r$  to  $Q_v$ , for a total of at

---

**Algorithm 5** Undirected.

---

**input** : Graph  $G(V, E)$  with  $t$  terminals  $S$ , and a number  $k$ .  
**output** : A Steiner  $k$ -tree

- 1  $R \leftarrow \{r\}$ ,  $S' \leftarrow S$ .
- 2 **while**  $k > 0$  **do**
- 3     Apply Procedure **Small** with  $\rho = t^{1/3}$  on  $C = V \setminus R$ . If it succeeds, return the resulting tree.
- 4     If **Small** fails, contract the terminals from each  $T_i$  in the resulting packing into a corresponding super-terminal  $q_i$ .
- 5     **if**  $C = V \setminus R$  is not a  $t^{1/3}$ -packing with respect to  $\{q_i\}$  **then**
- 6         Find a large tree  $Q_v$  inside  $G[C]$ .
- 7         Compute a shortest path  $P$  from  $r$  to  $Q_v$ .
- 8          $R \leftarrow R \cup P \cup Q_v$ .
- 9         Remove from  $S'$  all the terminals of  $Q_v$  and update  $k$ .
- 10    **else** *//  $C = V \setminus R$  is a  $t^{1/3}$ -packing.*
- 11
- 12     Apply Procedure **PMCover** with  $A = R$  and  $C = V - R$  with the goal of covering super-terminals.
- 13     Let  $\mathcal{E}$  be the edges corresponding to the returned sets  $(a, c) \in \mathcal{E}'$ , and  $T_C$  the shortest path tree on the corresponding trees  $T(c)$  (as in line 3 of **Complete**). Write  $Q = \mathcal{E} \cup T_C$ .
- 14      $R \leftarrow R \cup Q$ .
- 15     Remove from  $S'$  all the terminals  $\bigcup_i T_i$  and update  $k$ .

16 **return** the tree induced by  $R$

---

most  $2 \cdot t^{1/3} + 2$ . In the second case, the degree increases by at most  $t^{1/3}$  from  $T_C$  and 1 from  $\mathcal{E}$ , by an identical argument to Claim 4.5 (the proof of correctness of Procedure **Complete**).

In both cases, the vertex is immediately contracted into  $r$ .  $\triangleleft$

$\triangleright$  **Claim 5.8.** At every iteration, the degree of vertices in  $R$  is increased by at most  $O(\log k) \cdot B^*$ .

*Proof.* If a large tree  $Q_v$  is found, a shortest path from  $r$  to  $Q_v$  is computed. This increases the degree of any vertex by at most 2. Otherwise, Procedure **PMCover** is applied. This increases the degree of vertices of  $R$  by  $O(\log k) \cdot B^*$ . The claim follows.  $\triangleleft$

$\triangleright$  **Claim 5.9.** The returned tree contains  $k$  terminals, has maximum degree  $\tilde{O}(t^{1/3}) \cdot B^*$  and diameter  $O(D^*)$

*Proof.* By Claim 5.7, an iteration of Procedure **Undirected** increases the degree of a vertex  $v \notin R$  by at most  $O(t^{1/3})$ , any  $v$  whose degree increases immediately joins  $R$ . Now we bound the degree added to a vertex in  $R$ . By Claim 5.8 at every iteration the degree of  $v \in R$  can increase by  $O(\log k) \cdot B^*$ . By Claim 5.6, the number of iterations of Procedure **Undirected** is bounded by  $t^{1/3}$ . Therefore the total degree of a vertex is at most

$$O(t^{1/3}) + O(\log k) \cdot B^* \cdot t^{1/3} = O(\log k) \cdot t^{1/3} \cdot B^*.$$

In addition, the diameter of every  $Q_v$  or  $Q$  found, is  $O(D^*)$ . The distance of  $r$  to any  $Q$  or  $Q_v$  is at most  $D^*$  as well. This assures that the diameter is  $O(D^*)$ .

Finally, we argue that  $k$  terminals are covered. Fix a particular iteration of the algorithm. If Procedure `Small` succeeds in finding a  $t^{1/3}$ -additive partition, then we immediately cover the remaining number of terminals necessary by applying the Procedure `Complete`. Otherwise, we argue that among the terminals discarded in this iteration, the algorithm covers at least as many as  $T^*$  covers. Indeed, if  $C$  is not a  $t^{1/3}$ -packing with respect to  $\{q_i\}$ , then all terminals discarded are covered. On the other hand, if  $C$  is a  $t^{1/3}$ -packing, then by applying Procedure `PMCover`, Claim 5.5 ensures that at least  $\mu$  super-terminals are covered. Hence at least  $\mu \cdot t^{1/3} \geq k'$  terminals are covered, which is precisely the number of terminals covered by  $T^*$  among those discarded.  $\triangleleft$

Using [18] we get the following corollary that proves Theorem 1.3.

► **Corollary 5.10.** *The Minimum Time Telephone  $k$ -Multicast problem on undirected graphs admits a polynomial time,  $\tilde{O}(t^{1/3})$ -approximation algorithm.*

---

## References

- 1 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.
- 2 Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 1–12. ACM, 1987. doi:10.1145/41840.41841.
- 3 M. Elkin and G. Kortsarz. An approximation algorithm for the directed telephone multicast problem. *Algorithmica*, 45(4):569–583, 2006. doi:10.1007/s00453-005-1196-4.
- 4 Michael Elkin and Guy Kortsarz. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. *SIAM J. Comput.*, 35(3):672–689, 2005. doi:10.1137/S0097539704440740.
- 5 Michael Elkin and Guy Kortsarz. Sublogarithmic approximation for telephone multicast. *J. Comput. Syst. Sci.*, 72(4):648–659, 2006. doi:10.1016/j.jcss.2005.12.002.
- 6 M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. *An analysis of approximations for maximizing submodular set functions – II*, pages 73–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978. doi:10.1007/BFb0121195.
- 7 D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. *Technical report, UCLA/CSD-TR 02*, 2002.
- 8 Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, pages 319–349, 1988. doi:10.1002/net.3230180406.
- 9 R. Impagliazzo and R. Paturi. On the complexity of  $k$ -sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.
- 10 Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *J. Comput. Syst. Sci.*, 79(5):725–736, 2013. doi:10.1016/j.jcss.2013.01.019.
- 11 Guy Kortsarz and Zeev Nutov. The minimum degree group steiner problem. *Discret. Appl. Math.*, 309:229–239, 2022. doi:10.1016/j.dam.2021.12.003.
- 12 Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in a social communication network. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 435–443, 2008. doi:10.1145/1401890.1401945.

## 21:16 The Telephone $k$ -Multicast Problem

- 13 D. R. Kowalski and A. Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Comput.*, 19(3):185–195, 2007. doi:10.1007/s00446-006-0007-8.
- 14 Afshin Nikzad and R. Ravi. Sending secrets swiftly: Approximation algorithms for generalized multicast problems. In *ICALP*, pages 568–607, 2014. doi:10.1007/978-3-662-43951-7\_48.
- 15 Melih Onus and Andréa W. Richa. Minimum maximum-degree publish-subscribe overlay network design. *IEEE/ACM Trans. Netw.*, 19(5):1331–1343, 2011. doi:10.1109/TNET.2011.2144999.
- 16 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.
- 17 Da Qi Chen, Lin An, Aidin Niaparast, R Ravi, and Oleksandr Rudenko. Timeliness through telephones: Approximating information freshness in vector clock models. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2411–2428. SIAM, 2023. doi:10.1137/1.9781611977554.ch93.
- 18 R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time (extended abstract). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 202–213. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365693.

# Scheduling Splittable Jobs on Configurable Machines

Matthew Casey ✉ 

Northeastern University, Boston MA 02115, USA

Rajmohan Rajaraman ✉ 

Northeastern University, Boston MA 02115, USA

David Stalfa ✉

Northeastern University, Boston MA 02115, USA

Cheng Tan ✉ 

Northeastern University, Boston MA 02115, USA

---

## Abstract

Motivated by modern architectures allowing for the partitioning of a GPU into hardware separated instances, we initiate the study of scheduling splittable jobs on configurable machines. We consider machines that can be configured into smaller instances, which we call blocks, in multiple ways, each of which is referred to as a configuration. We introduce the Configurable Machine Scheduling (CMS) problem, where we are given  $n$  jobs and a set  $C$  of configurations. A schedule consists of a set of machines, each assigned some configuration in  $C$  with each block in the configuration assigned to process one job. The amount of a job's demand that is satisfied by a block is given by an arbitrary function of the job and block. The objective is to construct a schedule using as few machines as possible. We provide a tight logarithmic factor approximation algorithm for this problem in the general setting, a factor  $(3 + \varepsilon)$  approximation algorithm for arbitrary  $\varepsilon > 0$  when there are  $O(1)$  input configurations, and a polynomial time approximation scheme when both the number and size of configurations are  $O(1)$ . Finally, we utilize a technique for finding conic integer combinations in fixed dimension to develop an optimal polynomial time algorithm in the case with  $O(1)$  jobs,  $O(1)$  blocks, and every configuration up to a given size.

**2012 ACM Subject Classification** Theory of computation → Scheduling algorithms

**Keywords and phrases** Scheduling algorithms, Approximation algorithms, Configurable machines, Splittable jobs, Linear programming

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.22

**Category** APPROX

**Related Version** *Previous Version:* <https://doi.org/10.48550/arXiv.2312.05416>

**Funding** *Matthew Casey:* Partially supported by NSF grant CCF-1909363.

*Rajmohan Rajaraman:* Partially supported by NSF grants CCF-1909363 and CCF-2335187.

*David Stalfa:* Partially supported by NSF grant CCF-1909363.

*Cheng Tan:* Partially supported by NSF CAREER Award #2237295

## 1 Introduction

As the size of Deep Neural Network (DNN) models (particularly Large Language Models) continue to increase, there is a growing need to more efficiently allocate computational resources to these models at inference time. One challenge in efficiently allocating resources is that certain large models may require powerful GPUs, while other smaller models would greatly underutilize the power of such GPUs. To combat this issue, modern GPUs (e.g.,



© Matthew Casey, Rajmohan Rajaraman, David Stalfa, and Cheng Tan;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 22; pp. 22:1–22:20



Leibniz International Proceedings in Informatics

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

NVIDIA A30, A100, H100) include a new hardware feature called Multi-Instance GPU (MIG). MIG enables a GPU to be partitioned into smaller hardware isolated GPU instances, each with their own processors, memory, L2 cache, and bus bandwidth.

While MIG theoretically allows GPUs to avoid wasting resources, the feature raises the problem of efficiently scheduling on MIG-enabled GPUs. The problem presents two main challenges which must be considered simultaneously, and so compound the complexity of finding a solution. The first challenge is to partition the GPU into a configuration of smaller, variably sized GPU instances that can be used to execute DNN models. The second challenge is to assign models to these instances based on their resource demands. The problem is further complicated by the facts that (a) different configurations of GPU instances may have varying levels of computational and memory resources, (b) the resources are non-fungible for DNN models in the sense that increasing the size of a GPU instance may not linearly increase its performance [17], and (c) due to hardware constraints, some partitions of the GPU may not be available [13].

No prior work has provided algorithms with provable performance guarantees in the presence of (a-c), and currently deployed scheduling algorithms ignore either (b) [14], or (c) [16], or all three [19]. While this problem has gained much attention [17, 10, 11], these investigations primarily rely on heuristics with no formal guarantees. This paper is the first to establish theoretical bounds for scheduling on MIG-enabled GPUs.

We provide a natural formalization of the above problem, initiating a systematic theoretical study of scheduling splittable jobs in configurable machines. We call this problem *Configurable Machine Scheduling* or CMS. We consider machines that can be partitioned into multiple *configurations* of smaller instances, which we call *blocks*. We consider jobs that have certain demands that need to be satisfied by allocating blocks to it. Each job also has a corresponding table that specifies how much of the job’s demand is satisfied by a given block type. A *schedule* specifies each machine’s configuration and which job each of the machine’s blocks is to execute. Our goal is to construct a schedule that satisfies all job demands using as few machines as possible.

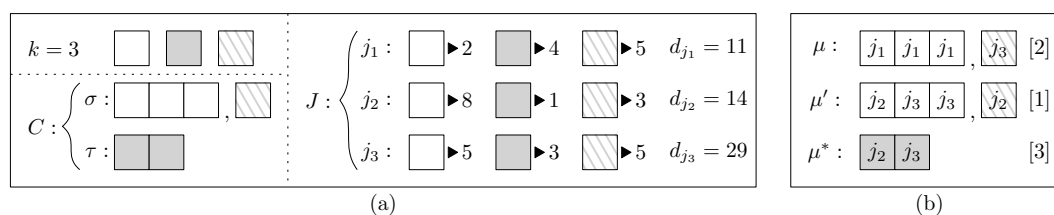
**Configurable Machine Scheduling (CMS).** A CMS instance is defined by a set  $C$  of *machine configurations* and a set  $J$  of *jobs*, as well as an integer  $k$  indicating the number of available *block types*.

- Each machine configuration  $\sigma \in C$  is a multiset of blocks represented as a length  $k$  vector. For  $i \in [k] = \{1, \dots, k\}$ , we let  $\sigma_i \in \mathbb{Z}_{\geq 0}$  indicate the number of blocks of type  $i$  in  $\sigma$ .
- For each job  $j \in J$ , there is an associated *demand*  $d_j$  as well as a length  $k$  *throughput table*  $f_j$ . For a job  $j \in J$  and block type  $i \in [k]$ , the value of  $f_j(i)$  denotes the amount of  $j$ ’s demand satisfied when it is executed on a single block of type  $i$ .

The goal is to satisfy all job demands on as few *machines* as possible. A machine  $\mu$  specifies how each block is allocated for each job. Specifically,  $\mu(i, j)$  represents the number of blocks of type  $i$  on which machine  $\mu$  executes job  $j$ , with the constraint that, there exists a configuration  $\sigma$  such that  $\sum_j \mu(i, j) \leq \sigma_i$ , for each  $i \in [k]$ . That is, each machine has an implicit configuration, and the total number of blocks of type  $i$  used by machine  $\mu$  cannot exceed the number of blocks of type  $i$  included in  $\mu$ ’s configuration.

A *schedule* consists of a multiset  $M$  of machines, with  $M_\mu$  indicating the number of instances of machine  $\mu$  in  $M$ . In any schedule, all job demands must be completely satisfied, i.e. for any schedule  $M$ , we require  $\sum_{\mu \in M} \sum_{i \in [k]} M_\mu \cdot \mu(i, j) \cdot f_j(i) \geq d_j$  for each job  $j$ . The formal objective is, then, to construct a schedule  $M$  that minimizes the total number of machine instances, given by  $\sum_{\mu} M_\mu$ . (See Figure 1 for an example.)





**Figure 1** Example CMS instance (a) with schedule (b). The instance has  $k = 3$  block types,  $|C| = 2$  configurations, and  $|J| = 3$  jobs. Configuration  $\sigma$  has three type 1 blocks, and one type 3 block. Configuration  $\tau$  has two type 2 blocks. Job  $j_1$  had demand 11, with throughput  $f_{j_1}(1) = 2$ ,  $f_{j_1}(2) = 4$ , and  $f_{j_1}(3) = 5$ . Job  $j_2$  had demand 14, with throughput  $f_{j_2}(1) = 8$ ,  $f_{j_2}(2) = 1$ , and  $f_{j_2}(3) = 3$ . Job  $j_3$  had demand 29, with throughput  $f_{j_3}(1) = 5$ ,  $f_{j_3}(2) = 3$ , and  $f_{j_3}(3) = 5$ . The schedule uses six machines: two instances of  $\mu$ , one instance of  $\mu'$ , and three instances of  $\mu^*$ .  $\mu$  and  $\mu'$  have configuration  $\sigma$ .  $\mu$  executes  $j_1$  on three type 1 blocks and  $j_3$  on one type 3 block.  $\mu'$  executes  $j_2$  on one type 1 block and one type 3 block, and  $j_3$  on two type 1 blocks. Machine  $\mu^*$  has configuration  $\tau$  and executes  $j_2$  on one type 2 block and  $j_3$  on the other. Summing the throughput over all machines is sufficient to satisfy all job demands.

In any CMS instance, we assume that, for all  $j$ ,  $f_j$  maps to  $\{\ell \in \mathbb{N} : \ell \leq d_j\}$ , which we can ensure with only a polynomial increase in the length of the input, and no loss in the value of the optimal solution. The *size* of a configuration  $\sigma$ , denoted  $|\sigma|$ , is the number of blocks in the configuration, i.e.  $|\sigma| = \sum_{i \in [k]} \sigma_i$ . Furthermore, for convenience, some of our algorithms output the schedule as a multiset of configurations, one for each machine in the output schedule, and a multiset of blocks for each job. In the appendix, we prove that this format of the output is without loss of generality, since it can be efficiently transformed to a schedule as formally defined above.

## 1.1 Our results

Our CMS problem formulation yields a rich landscape of optimization problems, which vary depending on the properties of block types, configurations, and the job demand tables. In this paper, we explore the general CMS problem and three restricted versions of the problem. We obtain near-tight approximations or optimal results for the associated problems (see Table 1).

**Table 1** Results for Configurable Machine Scheduling.  $n$  is the number of jobs,  $k$  is the number of block types, and  $c = \max_{\sigma \in C} \{|\sigma|\}$  is the maximum size of any configuration. All results are proved in this paper. The hardness of CMS with  $O(1)$  jobs and  $O(1)$  configuration size is unknown.

| Problem  | Algorithm                                     | Approximation  | Hardness          |
|--|---|--|-------------------|
| General  | LP+ Greedy                                    | $O(\log cnk)$  | $\Omega(\log nk)$ |
| $O(1)$ configurations  | Extreme-Point LP Rounding                     | $(2 + \varepsilon)\text{OPT} +  C $<br>$3 + \varepsilon$ | 2                 |
| $O(1)$ configurations of $O(1)$ size   | Small/Large Job LP                            | $1 + \varepsilon$  | ?                 |
| $O(1)$ number of jobs and blocks, with all configurations up to a given size | Conic Integer Combinations in Fixed Dimension | 1  | -                 |

**General CMS (Section 2).** Using a reduction from minimum multiset multicover [15], we first observe that CMS is hard to approximate to within a factor of  $\Omega(\log nk)$ , where  $n$  is the number of jobs and  $k$  the number of blocks. We then present a factor  $O(\log(cnk))$  approximation algorithm, where  $n$  is the number of jobs,  $k$  the number of blocks, and  $c$  is the size of the largest configuration, which is essentially tight given the above hardness result. Our algorithm constructs a schedule by greedily selecting the highest throughput configuration on the basis of a linear programming relaxation.

The logarithmic-hardness result for the general problem motivates us to consider restricted versions with a constant number of configurations, which are also of practical interest.

**CMS with a constant number of configurations (Section 3).** Using a reduction from Partition, we observe that CMS, *even with one configuration and two jobs*, is hard to approximate to within a factor of 2. Our main result is an algorithm that, for any instance of CMS with a constant number of configurations  $C$  and arbitrary  $\varepsilon > 0$ , uses at most  $(2 + \varepsilon)\text{OPT} + |C|$  machines where OPT is the number of machines needed in the optimal solution, asymptotically almost matching the hardness result for a constant number of configurations. We also show that our algorithm always returns a  $3 + \varepsilon$  approximation. Our algorithm builds on the seminal LP rounding technique of [9] and exploits the structure of extreme-point solutions to iteratively and carefully round the LP variables.

To find more tractable cases, we study a further restriction of the problem that bounds the size of configurations.

**CMS with a constant number of configurations of constant size (Section 4).** We next consider CMS with a constant number of configurations, each of constant size (i.e., having a constant number of blocks). We show that the problem is solvable in pseudo-polynomial time; our main result here is a PTAS based on rounding a novel LP relaxation for the problem.

Our LP based approximations require the number of configurations given in the input to be constant. Our final result explores nontrivial tractable models where the number of jobs is constant while the number and size of configurations are not constant.

**CMS with a constant number of jobs and blocks, with all configurations up to a given size (Section 5).** We consider CMS with a constant number of jobs, a constant number of block types, and where every configuration up to a given size is available. We give an algorithm that solves this problem optimally in polynomial time. Our algorithm uses a technique for finding conic integer combinations in fixed dimension, which was developed by Goemans and Rothvoss in their study of bin packing with constant number of job types [5]. We also show that this technique extends to a more general setting where configurations are defined by a constant number of rational polytopes.

## 1.2 Our Techniques

**General CMS.** The logarithmic factor approximation algorithm for the most general case uses two algorithmic approaches, each of which faces challenges on its own. The first approach uses the natural heuristic of greedily allocating machines to execute as much total throughput as possible. When each job can be fully executed on at most one block of each type, we

show that this algorithm achieves a logarithmic approximation ratio. However, in general the algorithm's approximation ratio is  $\Omega(n)$ . The second approach is based on an LP relaxation that uses job-block variables to indicate how many blocks of a given type are used to execute a job. When the job-block variables are large enough (greater than 1) they can be rounded and scheduled using a multi-set multi-cover algorithm to achieve a logarithmic approximation ratio. On the other hand, variables with low values cannot be rounded up without a large loss in the approximation ratio, and cannot be rounded down because even a small fractional block allocation may represent a significant amount of *throughput*. Our solution is to marry these approaches. We first solve the LP relaxation and use a multi-set multi-cover algorithm to schedule those job-block pairs with large variable values. For any jobs with remaining demand, we show that they can be fully executed on at most one block of each type, and ensure that the maximum ratio of execution function values is at most the number of block types  $k$ . Executing the greedy algorithm on these remaining jobs yields an algorithm with a logarithmic approximation ratio.

**Constant number of configurations.** In this case, we use essentially the same LP as for the general problem, simplifying it to use only variables that represent how many of each block type are allocated to a job. We then build a graph with nodes representing the block types and jobs, and insert an edge between a job and block type if the corresponding variable in the LP is nonzero. We leverage extreme point properties to prove that the graph is a pseudo-forest; i.e., each component is either a tree or a tree with a cycle. Our key technical contribution is to carefully round the LP solution by exploiting this tree structure and the constraints on the possible LP values mandated by the fact that, by our construction, no block can satisfy more demand than the job requires. This algorithm returns a  $(2 + \varepsilon)\text{OPT} + |C|$  approximation. We extend this result to a  $(3 + \varepsilon)\text{OPT}$  approximation (which is better for  $\text{OPT} < |C|$ ) by running the above algorithm on each subset of the input configurations, and then returning the best solution.

**Constant number of configurations of constant size.** The LP used in the preceding two variants of the problem has an integrality gap of 2. This holds even for simple instances with one configuration of constant size and two jobs. To obtain a PTAS for a constant number of configurations of constant size, we first divide the jobs into small and large jobs based on whether the number of machines needed to serve their demand exceeds a constant threshold (based on a parameter  $\varepsilon$ ). We then formulate a new LP that imposes different constraints for the small jobs taking into account that there is a bounded number of ways their demands can be allocated. Through a careful rounding of the LP, we derive a  $(1 + \varepsilon)$ -approximation algorithm for the problem.

**Constant number of jobs and blocks, and all configurations up to a given size.** In this setting neither the number of configurations nor the size of any configuration is constant, and so any approach based on our LP relaxations faces the obstacle that either the program has an integrality gap of at least 2, or is intractable. Therefore, in order to solve this problem optimally, more sophisticated techniques are required. The approach we develop is based on a technique for finding conic integer combinations in fixed dimension which was developed by Goemans and Rothvoss in their study of bin packing with constant job types [5]. The challenges of this approach lie in formulating the problem to appropriately leverage the power of the apparatus. In our case, this involves providing a rational representation of the problem on an individual machine that yields a complete solution when combined with the

representations of other machines. In the general case, the nonlinear relationship of blocks to job throughput *and* to configuration size renders the problem extremely difficult, even when jobs and blocks are constant. However, when all configurations up to a given size are available, a configuration can be represented as a linear combination of block allocations, making the problem tractable. In fact, as long as the number of jobs and blocks types is constant, the general technique for finding conic integer combinations allows us to devise an optimal polynomial time algorithm whenever the set of configurations can be represented using a constant number of rational polytopes.

### 1.3 Related work

Configurable machine scheduling has connections to many well-studied problems in combinatorial optimization, including bin-packing, knapsack, multiset multicover, and max-min fair allocation. General CMS generalizes the multiset multicover problem [8, 6, 15], for which the best approximation factor achievable in polynomial time is  $O(\log m)$  where  $m$  is the sum of the sizes of the multisets [15, 18]. The hardness of approximating the problem to within an  $O(\log n)$  factor follows from the result for set cover [4].

As we note above, CMS is NP-complete even for the case of one configuration and two jobs. The single configuration version can be viewed as a fair allocation problem with each block representing an item and each job representing a player that has a value for each item (given by the demand table) and a desired total demand. The objective is to minimize the maximum number of copies we need of each block so that they can be distributed among the players satisfying their demands. In contrast, the Santa Claus problem in fair allocation [1] (also studied under a different name in algorithmic game theory [12]) aims to maximize the minimum demand that can be satisfied with the available set of blocks. The best result for the Santa Claus problem is a quasi-polynomial  $O(n^\varepsilon)$ -approximation algorithm, where  $\varepsilon = O(\log \log n / \log n)$  [2], though factor  $O(1)$  approximation algorithms are known for special cases (e.g., see [3]).

### 1.4 Discussion and Open Problems

Our study has focused on a *combinatorial* version of CMS in which each machine can be configured as a collection of abstract blocks. It is also natural to consider a *numerical* version of CMS in which each block type is an item of a certain size, and each configuration has a certain capacity and can only fit blocks whose sizes add up exactly to its capacity. Note that instances of numerical CMS can be presented more compactly than general instances of CMS since the allowable configurations can be captured by configuration capacities and block sizes. The approximation ratios established for CMS apply to numerical CMS as well; however, it is not certain that there is also a logarithmic hardness for numerical CMS. Thus, an intriguing open problem is whether numerical CMS admits an approximation factor significantly better than the logarithmic factor established in Section 2. Also of interest is a numerical CMS variant where all capacity-bounded configurations are allowed, for which we believe techniques from unbounded knapsack [7] and polytope structure results of the kind we show in Section 5 would be useful.

Our results indicate several directions for future research. One open problem is to devise approximation algorithms that leverage structure in the set of available configurations. In practice, the configuration sets associated with multi-instancing GPUs might not be arbitrary sets, e.g. the blocks of Nvidia's A100 GPU are structured as a tree and every valid configuration is a set of blocks with no ancestor-descendant relations [17]. Showing improved bounds for such cases seems to be a challenging, but potentially fruitful area of research.

Another open problem lies in shrinking the gap between our upper and lower bounds. The hard instance for CMS with a constant number of configurations has a constant-size solution, showing for instance that it is NP-hard to distinguish a problem with solution size 1 from one with solution size 2. These lower bounds are sufficient to show hardness of approximation, but do not rule out the possibility of asymptotic PTAS (even additive constant approximations). Furthermore, we have not been able to show any hardness for CMS with a constant number of configurations of constant size, and this is an important and interesting open problem.

Finally, our focus has been on the objective of minimizing the number of machines, which aims to meet all demands using minimum resources. Our results can be extended to minimizing makespan, given a constant number of machines. However, approximations for other objectives such as completion time or flow time, in both offline and online settings, are important directions for further research.

## 2 General CMS logarithmic approximation

In this section, we consider the most general model of CMS with an arbitrary configuration set  $C$  over  $k$  block types, and  $n$  jobs in  $J$ . Our first lemma presents an approximation-preserving reduction from multiset multicover to CMS. We thus obtain that no polynomial time algorithm can achieve an approximation ratio better than  $\Omega(\log nk)$  (assuming  $P \neq NP$ ). The lemma also implies that an improvement to our approximation ratio would yield an improvement to the best known approximation for multiset multicover.

► **Lemma 1.** *There is an approximation-preserving reduction from the multiset multicover problem to CMS.*

**Proof.** Consider an arbitrary instance  $I$  of multiset multicover. Let  $\mathcal{U}$  denote the set of elements and  $\mathcal{C}$  the collection of multisets in the set cover instance. Let  $r_e$  denote the coverage requirement for element  $e$ . We can assume without loss of generality that there do not exist two multisets  $S$  and  $S'$  with  $S \subseteq S'$ , since we can eliminate  $S$  from the set collection otherwise. We construct an instance of CMS where each multiset  $S$  is a configuration and each element  $e$  is both a block type and a job. The job  $e$  has demand  $r_e$ , which can only be satisfied by  $r_e$  blocks of type  $e$ .

Any multiset multicover solution, given by a collection  $M$  of multisets, corresponds to a solution for CMS: each multiset  $S$  in  $M$  is a machine configured according to  $S$ . Therefore, the number of multisets in  $M$  is the same as the number of machines in the CMS solution. Furthermore, since each element  $e$  is covered  $r_e$  times in  $M$ , it follows that each job  $e$  has  $r_e$  occurrences of block type  $e$  included in CMS solution, thus satisfying the demand for  $e$ . Similarly, every CMS solution with  $m$  machines is a collection of  $m$  multisets, with each multiset corresponding to the configuration of a machine. Since the objective function value achieved by each of the two solutions is identical, the reduction is approximation-preserving. ◀

The main result of this section is Algorithm 1, an  $O(\log(\max_{\sigma \in C} \{|\sigma|\} \cdot n \cdot k))$ -approximation algorithm for CMS. The first step of Algorithm 1 solves a linear relaxation of CMS, which minimizes  $\sum_{\sigma} y_{\sigma}$  subject to:

$$\sum_j x_{i,j} \leq \sum_{\sigma \in C} y_{\sigma} \cdot \sigma_i \quad i \in [k] \tag{1}$$

$$\sum_i f_j(i) \cdot x_{i,j} \geq d_j \quad j \in J \tag{2}$$

$$x_{i,j} \geq 0 \quad i \in [k] \text{ and } j \in J \tag{3}$$

$$y_{\sigma} \geq 0 \quad \sigma \in C \tag{4}$$

■ **Algorithm 1** Logarithmic Approximation for CMS.

- 
- 1 *Formulate and Solve a Linear Relaxation (Constraints 1-4)*  
Round variables down if their fractional component is less than  $(1/2k)$
  - 2 *Solve Problem over the Integer Components of Variables (Algorithm 2)*  
Construct partial schedule  $S$  via multiset-multicover defined over variable integer components
  - 3 *Greedily Schedule any Jobs with Remaining Demand (Algorithm 3)*  
Construct a partial schedule  $S'$  to satisfy any remaining demand by greedily configuring each machine to maximize throughput
  - 4 *Output the schedule formed by the additive union  $(S \oplus S) \oplus (S' \oplus S')$*
- 

**Terms and Constraints.** Each variable  $x_{i,j}$  indicates the number of blocks of type  $i$  that are assigned to execute job  $j$ . Each variable  $y_\sigma$  indicates the number of machines that use configuration  $\sigma$ . Constraint 1 ensures a schedule cannot use more blocks of a given type than appear across all allocated machines. Constraint 2 states that the total number of blocks executing a job must be sufficient to satisfy its demand. It is easy to verify that this program relaxes CMS and is solvable in polynomial time.

Let  $(x^*, y^*)$  be variable assignments that yield an optimal solution to (1-4). For the second step of Algorithm 1, we separate the integer from the fractional components of the  $x$ -variables. We define  $\bar{x}_{i,j} = \lfloor x_{i,j}^* \rfloor$ .

We define  $\hat{x}_{i,j} = 0$  if either (i)  $(x_{i,j}^* - \lfloor x_{i,j}^* \rfloor) < \frac{1}{2k}$  or (ii)  $f_j(i) \cdot (x_{i,j}^* - \lfloor x_{i,j}^* \rfloor) < \max_{i'} \{f_j(i') \cdot (x_{i',j}^* - \lfloor x_{i',j}^* \rfloor)\} / k$ , otherwise  $\hat{x}_{i,j} = x_{i,j}^* - \lfloor x_{i,j}^* \rfloor$ . The second step of Algorithm 1 then calls Algorithm 2 to provide a schedule for the problem  $(C, \bar{J}, k)$  with modified demands  $\bar{d}_j = \min\{d_j, \sum_i f_j(i) \cdot \bar{x}_{i,j}\}$ .

**Algorithm 2.** We define the set  $A = \{(\sum_j \lfloor x_{i,j} \rfloor, i) : i \in [k]\}$ . We construct schedule  $S$  by using the greedy multiset multicover algorithm given in [15] on the instance  $(A, C)$ .

Step three of Algorithm 1 then constructs a schedule  $S'$  to satisfy any remaining demand given by the fractional components  $\hat{x}$  via Algorithm 3, which greedily allocates the highest throughput machines until all demands are met. Finally, Algorithm 1 outputs the schedule  $S^*$  such that,  $S_\mu^* = 2(S_\mu + S'_\mu)$  for each  $\mu$ .

**Algorithm 3.** For each  $j \in J$  and  $i \in [k]$ , we initialize  $\hat{f}_j(i) = f_j(i)$ . While there is some job that hasn't been fully executed, do the following. Compute a machine  $\mu_{\max}$  by iterating over all configurations  $\sigma$  and greedily allocating each block of  $\sigma$  to the job with the highest throughput for that block (accounting for demand already satisfied). Of these machines, greedily set  $\mu_{\max}$  to be the one with highest total throughput. Allocate enough instances of  $\mu$  such that some job's remaining demand becomes less than  $\hat{f}_j(i)$  for some  $i$ . Then, for each  $j, i$ , update  $\hat{f}_j(i) \leftarrow \min\{\hat{f}_j(i), D_j\}$  where  $D_j$  is the job's remaining demand. Then repeat.

In the remainder of the section, we provide analysis of Algorithm 1. The following lemmas establish bounds on the lengths of the schedules produced by Algorithm 3. We note that there exist instances for which Algorithm 3 produces schedules with length  $\Omega(n)$ , and so it cannot be used to solve CMS without some additional processing (which, for us, involves reducing the instance via Algorithm 2).

► **Lemma 2.** *If  $d_j \leq \sum_i f_j(i)$  for all  $i$  (i.e. each job can be executed on at most one block of each type) then Algorithm 3 returns an  $O(\log(\max_{\sigma \in C}\{|\sigma|\} \cdot nk\rho))$  approximation, where  $\rho = \max_{i,i',j}\{f_j(i)/f_j(i')\}$ .*

**Proof.** Consider the following integer program formulation of CMS, which minimizes  $\sum_{t,\sigma} w_{t,\sigma}$  subject to:  $\sum_{t,\sigma,i} z_{t,i,j} \cdot f_j(i) \geq d_j$ ,  $j \in J$  and  $\sum_j z_{t,i,j} \leq w_{t,\sigma} \cdot \sigma_i$ ,  $t \in \mathbb{N}$ ,  $\sigma \in C$ ,  $i \in [k]$  and  $\sum_{\sigma} w_{t,\sigma} \leq 1$ ,  $t \in \mathbb{N}$  and  $w_{t,\sigma}, z_{t,\sigma,i,j} \in \mathbb{N}$ ,  $t \in \mathbb{N}$ ,  $\sigma \in C$ ,  $i \in [k]$ ,  $j \in J$ . In this program,  $w_{t,\sigma} = 1$  if the  $t$ th machine instance has configuration  $\sigma$  (0 otherwise),  $z_{t,i,j}$  = the number of blocks of type  $i$  that the  $t$ th machine uses to execute job  $j$ . It is easy to see that the program relaxes CMS.

Let  $(w^*, z^*)$  be a variable assignment that minimizes the objective with value  $\eta$ . Then

$$\sum_j d_j \leq \sum_{\mu \in S} \mu(i,j) \cdot f_{\mu(i)}(i) \leq \eta \cdot k \cdot \max_{\sigma \in C} \{|\sigma|\} \cdot \max_{i,j} \{f_j(i)\}$$

The fact that  $d_j \leq \sum_i f_j(i)$  for all  $j$  implies that  $\eta \leq nk$  since, in the worst case, each block is executed on its own machine. So we can infer that  $\log(\sum_j d_j) \leq 2 \cdot \log(nk \cdot \max_{\sigma \in C} \{|\sigma|\} \cdot \max_{i,j} \{f_j(i)\})$ . In the remainder of the proof, we show that the greedy algorithm has an approximation ratio of  $\log(\sum_j d_j)$ .

Let  $(\bar{w}, \bar{z})$  be the variable assignment given by schedule produced by Algorithm 3. Let  $\bar{u}_t = \sum_{\sigma,i,j} \bar{z}_{t,\sigma,i,j}$  for every  $t$ . We define  $u_t^*$  and  $v_t^*$  to be any values that satisfy the following equalities:  $u_t^* + v_t^* = \sum_{\sigma,i,j} f_j(i) z_{t,\sigma,i,j}^*$ ,  $t \in \mathbb{N}$  and  $\sum_t v_t^* = \sum_j d_j - \sum_{t \leq \eta} \sum_{\sigma,i,j} f_j(i) \bar{z}_{t,\sigma,i,j}$  and  $\sum_t u_t^* = \sum_j d_j - \sum_t v_t^*$ . Informally,  $\bar{u}_t$  is amount of throughput achieved by machine instance  $t$  of  $(\bar{w}, \bar{z})$ .  $u_t^*$  is the amount of throughput achieved by machine instance  $t$  of  $(w^*, z^*)$  that is also satisfied by one of the first  $\eta$  machines in  $(\bar{w}, \bar{z})$ .  $v_t^*$  is the remaining demand satisfied by machine instance  $t$  of  $(w^*, z^*)$ . It is easy to see that,  $\sum_{t \leq \eta} \bar{u}_t = \sum_t u_t^*$ . We show that  $2 \sum_{t \leq \eta} \bar{u}_t \geq \sum_{t \leq \eta} v_t^*$ . Suppose, for the sake of contradiction, that the claim is false. Then for some  $t$ ,  $2\bar{z}_{\tau+1,\sigma,i,j} < v_t^*$ . However, since  $v^*$  represents demand not satisfied by  $(\bar{w}, \bar{z})$  in the first  $\eta$  machines, and since the machine  $\mu_{\max}$  in Algorithm 3 has throughput at least half of the maximum (proved below), in this case the throughput of  $\mu_{\max}$  would be at least  $v_t^*$ . This is a contradiction.

We now show that, when Algorithm 3 constructs  $\mu_{\max}$ , its throughput is at least half the throughput of the highest throughput machine possible. Let  $\sigma$  be the configuration used by  $\mu_{\max}$ . We show that the maximum throughput machine  $\mu^*$  over  $\sigma$  has throughput no more than twice that of  $\mu_{\max}$ . We consider an integer program formulation of the problem of finding the maximum throughput machine:  $w_j \leq d_j$ ,  $j \in J$  and  $w_j \leq \sum_i z_{i,j} f_j(i)$ ,  $j \in J$  and  $\sum_j z_{i,j} \leq \sigma_i$ ,  $i \in B$ .

The set  $B$  includes all block instances in  $\sigma$ . Let  $(w^*, z^*)$  represent the solution to this program given by the optimal machine  $\mu^*$ . Let  $(\bar{w}, \bar{z})$  represent the solution to this program given by  $\mu_{\max}$ . For each block  $i$ , we set  $u_i$  and  $v_i$  to be any values that satisfy the following equations.  $u_i + v_i = \sum_j z_{i,j}^* f_j(i)$ ,  $i \in [k]$  and  $\sum_i v_i = \sum_j w_j^* - \sum_{i,j} \bar{z}_{i,j} f_j(i)$  and  $\sum_i u_i = \sum_j w_j^* - \sum_i v_i$ . Informally,  $u_i$  is the amount of demand satisfied by block  $i$  of  $\mu^*$  that is also satisfied by  $\mu_{\max}$ , and  $v_i$  is the amount of demand satisfied by block  $i$  of  $\mu^*$  that is not satisfied by  $\mu_{\max}$ . For any block  $i$ , it is easy to see that  $\sum_j \bar{z}_{i,j} \geq u_i$ . We can also infer that  $\sum_j \bar{z}_{i,j} \geq v_i$  because, otherwise, the greedy algorithm would have chosen to execute the same job as  $\mu$  on block  $i$ . This proves the result.

Since  $\sum_{t \leq \eta} \bar{u}_t = \sum_t u_t^*$  and  $4 \sum_{t \leq \eta} \bar{u}_t \geq \sum_t v_t^*$ , the total demand satisfied by  $(\bar{w}, \bar{z})$  is at least 1/4 the total demand. It is straightforward to generalize this reasoning to show that every  $\eta$  machine instances of  $(\bar{w}, \bar{z})$ , reduce the total demand by a factor of 1/4, which proves our claim. ◀

## 22:10 Scheduling Splittable Jobs on Configurable Machines

► **Theorem 3.** *Algorithm 1 returns an  $O(\log(\max_{\sigma \in C}\{|\sigma|\} \cdot n \cdot k))$  approximation to the CMS problem.*

**Proof.** Let  $S$  and  $S'$  represent the schedules produced by Algorithm 2 and Algorithm 3, respectively. We first argue that  $S$  has length  $O(\log(\max_{\sigma}\{|\sigma|\} \cdot n)) \cdot \text{OPT}$ . Algorithm 2 reduces scheduling the integer components of the variables to an instance of multi-set multi-cover in which there are  $n$  elements and the largest covering multi-set has size  $\max_{\sigma}\{|\sigma|\}$ . The claim follows directly from Theorem 5.1 in [15].

We now show that  $S'$  has length  $O(\log(\max_{\sigma}\{|\sigma|\} \cdot nk)) \cdot \text{OPT}$ . By Lemma 2, we need only to show that  $d_j \leq \sum_i f_j(i)$  and that  $\max_{j,i,i'}\{f_j(i)/f_j(i')\} = O(k)$ . We can infer  $d_j \leq \sum_i f_j(i)$  because  $(C, J, k)$  with modified demand tables and demands  $\hat{f}, \hat{d}$  is defined over  $\hat{x}$ , so each job can be completely executed by one block of each type. Also, the definition of  $\hat{x}$  entails that for each  $j$ , every nonzero value of  $\hat{x}_{i,j}$  (resp.  $\hat{f}_j(i) \cdot \hat{x}_{i,j}$ ) is within a factor of  $2k$  (resp.  $k$ ) of every other.

Finally, in defining  $\hat{x}$ , we rounded down  $x_{i,j}^*$  if (i)  $z_{i,j}^* < 1/2k$  or if (ii)  $z_{i,j}^* \cdot f_j(i) < \max_{i'}\{z_{i',j}^* \cdot f_j(i')\}/k$ . Job  $j$ 's total reduction in demand from (i) is no more than  $d_j \sum_i x_{i,j}^* - \bar{x}_{i,j} \leq d_j/2$ , which is accounted for by doubling  $S_1$  and  $S_2$  in the output. Job  $j$ 's total reduction in demand due to (ii) is at most  $\max_{i'}\{z_{i',j}^* \cdot f_j(i')\}$  which is accounted for in setting  $\hat{x}_{i,j} = 2z_{i,j}^*$  for all remaining  $i$ 's. Each doubles our approximation ratio. ◀

### 3 CMS with a constant number of configurations

We consider CMS with  $n$  jobs and a set  $C$  of  $O(1)$  configurations, each of arbitrary size. We first observe that the problem is NP-hard to approximate to within a factor of two.

► **Lemma 4.** *CMS with a constant number of configurations is hard to approximate to within a factor of 2.*

**Proof.** We present a reduction from Partition to combinatorial CMS. Given an instance of Partition with a set  $S$  of  $n$  elements  $0 < a_1 < a_2 < \dots < a_n$ , we construct the following instance. We consider one configuration that contains  $n$  blocks all of a different type, labeled  $1, \dots, n$ . We have two jobs  $j_1, j_2$ , both with the demand function given by  $f(i) = a_i$ . The demand for each job is  $\frac{1}{2} \sum_i a_i$ .

We claim that the number of machines needed for scheduling the job is one if and only if the Partition instance has a yes answer. If the Partition instance has a yes answer, then there exists a way to split the  $n$  blocks into two parts so that each part's value adds up to  $\sum_i a_i/2$ . We use one machine, and assign the blocks to each job according to the Partition solution. The demand function ensures that the demand of the job is satisfied. Conversely, if the demand of the two jobs is satisfied by one machine, then each job has at least  $\sum_i a_i/2$  demand satisfied. According to the demand function  $f$ ,  $\sum_i a_i$  is the maximum amount of demand that can be satisfied by this configuration. Thus each job has exactly  $\sum_i a_i/2$  demand satisfied, and so there is some partition of  $S$  into two parts such that each part sums to  $\sum_i a_i/2$ . ◀

Our main result in this section is a polynomial time algorithm that returns a solution with cost the minimum of  $(2 + \epsilon)\text{OPT} + |C|$  and  $(3 + \epsilon)\text{OPT}$ , for arbitrary  $\epsilon > 0$ , where OPT is optimal cost. Our algorithm, detailed in Algorithm 4, guesses the number of each configuration used in an optimal solution, to within a factor of  $1 + \epsilon$  (see line 3), and then builds on the paradigm of [9] by carefully rounding an extreme-point optimal solution for a suitable instantiation of LP(1-4) (given in line 5).



■ **Algorithm 4** Schedule for CMS with  $O(1)$  configurations.

---

**Input:** A CMS instance  $(C, J, k)$

- 1  $L \leftarrow \{ \lfloor (1 + \varepsilon)^i \rfloor \mid 0 \leq i \leq \log_{1+\varepsilon}(\sum_j d_j) \}$ ,  $Sol \leftarrow \{ \}$
- 2 **foreach**  $C^* \in P(C)$ , the powerset of  $C$  **do**
- 3     **foreach**  $m \in L^{|C^*|}$ , where  $m_\sigma$  is the number of copies of  $\sigma$  **do**
- 4          $B^* \leftarrow \{ i \in [k] \mid \exists \sigma \in C^* \text{ s.t. } i \in \sigma \}$  is the set of blocks present in  $C^*$
- 5         Construct the feasibility LP,  $LP_f$  with constraints from equations (1'), (2), and (3).
 
$$\sum_j x_{i,j} \leq \sum_{\sigma \in C^*} m_\sigma \cdot \sigma_i \quad \text{block types } i \in B^* \quad (1')$$
- 6         **if**  $LP_f$  is feasible with extreme-point solution  $x$  **then**
- 7             Graph  $G \leftarrow (J \cup B^*, E)$  with  $E = \{ (i, j) \mid x_{i,j} > 0 \}$
- 8             **foreach** Component  $S \in G$  that has a cycle  $K$  **do**
- 9                 Pick some job  $j$  in the cycle  $K$ , and let  $b_1, b_2$  be its neighbors in the cycle
- 10                 **if**  $x_{b_1,j} \cdot f_j(b_1) \geq x_{b_2,j} \cdot f_j(b_2)$  **then**  $E \leftarrow E \setminus \{ (b_2, j) \}$  **else**  
 $E \leftarrow E \setminus \{ (b_1, j) \}$
- 11                 Make  $j$  the root of the remaining tree  $S$
- 12             **foreach** Job  $j \in J$  **do**
- 13                 **for** the parent block  $p$  of  $j$ , **do**  $x_{p,j}^* \leftarrow \lfloor 2x_{p,j} \rfloor$
- 14                 **foreach** child block  $c$  of  $j$  **do**  $x_{c,j}^* \leftarrow \lceil 2x_{c,j} \rceil$
- 15             **foreach** Configuration  $\sigma \in C^*$  **do**  $y_\sigma^* \leftarrow 2m_\sigma + 1$
- 16             **if** fewer configurations are used in  $y^*$  than in  $Sol$  **then**  $Sol \leftarrow (x^*, y^*)$
- 17             **break out of iteration**

18 **return**  $Sol$  transformed from  $(x^*, y^*)$  format to a machine schedule according to Algorithm 6

---

Using extreme-point properties, we establish Lemma 5, the proof of which closely follows [9].

► **Lemma 5.** *Every component in graph  $G$  of line 7 has at most one cycle.*

**Proof.** This proof follows a similar structure as the proof of Lemma 17.6 in [18]. We will use a proof by contradiction. First, consider a component in  $G$ , called  $G_c$ . Then consider the restriction of the LP,  $LP_c$ , to only the jobs and block types present in the component. Also let  $x_c$  be the restriction of  $x$  to those jobs and blocks present in the component. Let  $x_{\bar{c}}$  be the rest of  $x$ . Note that  $x_c$  is a feasible solution to  $LP_c$  since all the blocks that satisfy demand for jobs in  $G_c$  are connected to those jobs in the original graph  $G$  and thus are also included in  $G_c$ , so we continue to satisfy all the demand for these jobs. Now assume for contradiction that  $x_c$  is not an extreme point in  $LP_c$ . Then  $\exists x_1, x_2, \lambda$  where  $x_1$  and  $x_2$  are feasible solutions to  $LP_c$  and  $\lambda \in (0, 1)$  such that we have  $x_c = \lambda \cdot x_1 + (1 - \lambda) \cdot x_2$ .

Now we show that  $x_1 + x_{\bar{c}}$  and  $x_2 + x_{\bar{c}}$  are feasible solutions to the LP. First, consider that  $x_1, x_2$  have disjoint jobs and block types from  $x_{\bar{c}}$ . Thus, we can consider their corresponding constraints separately. Furthermore, together,  $x_1, x_2$ , and  $x_c$  cover all the constraints (since they cover all jobs and block types). Thus we need only verify that  $x_1, x_2$  satisfy their

## 22:12 Scheduling Splittable Jobs on Configurable Machines

constraints, and  $x_{\bar{c}}$  satisfies its constraints. Since  $x_1, x_2$  are feasible solutions to  $LP_c$  we know they satisfy the constraints in  $LP$  relevant to them. And since  $x_{\bar{c}}$  is part of the feasible solution  $x$ , it must also satisfy the constraints relevant to it. Between the two, all the constraints of the  $LP$  are satisfied, since together they cover all jobs and blocks.

But then since  $x = \lambda \cdot (x_1 + x_{\bar{c}}) + (1 - \lambda) \cdot (x_2 + x_{\bar{c}})$  we can say that  $x$  is a convex combination of two other solutions. Thus,  $x$  is not an extreme point solution. But, since  $x$  is an extreme point solution, we reach a contradiction.

Therefore,  $x_c$  must be an extreme point solution in  $LP_c$ . We know that the number of tight constraints in an extreme point solution is at least as many as the number of variables. Thus at most  $|B_c^*| + |J_c|$  constraints can be not tight (coming from constraints 1' and 2). Since we create an edge only if an  $x$  variable is nonzero, or equivalently, its nonzero constraint is not tight, we know that the number of edges in  $G_c$  must be at most the number of blocks + jobs in  $G_c$ . In other words, the number of edges is at most the number of nodes. Since  $C$  was chosen arbitrarily, we conclude that every component in  $G$  has at most one cycle. ◀

► **Lemma 6.** *Algorithm 4 returns a feasible integer solution to LP(1-4).*

**Proof.** Since the algorithm returns the least cost rounded solution over all iterations, we need to show that  $(x^*, y^*)$  is a feasible integer solution to LP(1-4). By our rounding,  $x_{i,j}^*$  and  $y_{\sigma}^*$  are integers for each  $i, j, \sigma$ . It remains to show that  $(x^*, y^*)$  is feasible in LP(1-4). Constraints 3 and 4 are true by definition of  $x^*, y^*$ .

We now consider constraint 1. If a block type  $i$  is not in  $B^*$ , then this constraint is satisfied because  $x_{i,j} = 0$  for all  $j$ , and thus  $x_{i,j}^* = 0$  for all  $j$ . Now we consider blocks that are in  $B^*$ . By Lemma 5, each component of  $G$  has at most one cycle. In the algorithm we remove an edge from each of these cycles, so the resulting graph is a forest. Thus each block type  $i$  has one parent and so is a child of one job. This means that all  $x_{i,j}$  variables associated with block  $i$  are rounded as  $\lfloor 2x_{i,j} \rfloor$ , except for the parent of  $i$ ,  $p_i$ . So

$$\begin{aligned} \sum_j x_{i,j}^* &= \sum_{j \neq p_i} \lfloor 2x_{i,j} \rfloor + \lceil 2x_{i,p_i} \rceil \leq \sum_j 2x_{i,j} + 1 \leq \sum_{\sigma \in C^*} 2m_{\sigma} \cdot \sigma_i + 1 \\ &\leq \sum_{\sigma \in C^*} (2m_{\sigma} + 1) \cdot \sigma_i \leq \sum_{\sigma \in C} y_{\sigma}^* \cdot \sigma_i \end{aligned}$$

The third inequality follows from constraint 1' since  $x$  satisfies  $LP_f$ , and the fourth inequality holds since  $i \in B^*$  implying that there is at least one  $\sigma \in C^*$  with  $\sigma_i \geq 1$ . Thus, constraint 1 is satisfied.

Now, we consider constraint 2. First, we consider some job  $j$  whose edge was not removed. Then, since  $G$  becomes a forest after pruning edges we obtain that either the children or the parent of  $j$  satisfy at least half of its demand. If its children satisfy at least half of its demand then we have  $\sum_{\text{children of } j} f_j(i) \cdot x_{i,j} \geq \frac{1}{2}d_j$  and thus we obtain

$$\sum_i f_j(i) \cdot x_{i,j}^* \geq \sum_{\text{children of } j} f_j(i) \cdot \lfloor 2x_{i,j} \rfloor \geq 2 \sum_{\text{children of } j} f_j(i) \cdot x_{i,j} \geq d_j,$$

Therefore, the constraint is satisfied. Otherwise, its parent  $p$  satisfies at least half of its demand implying that  $x_{p,j} \geq \frac{1}{2}$  since we have  $f_j(p) \leq d_j$  by our assumption on the input. Then,  $x_{p,j}^* = \lfloor 2x_{p,j} \rfloor > x_{p,j}$ , yielding  $\sum_i x_{i,j}^* \cdot f_j(i) \geq \sum_i x_{i,j} \cdot f_j(i) \geq d_j$  since  $x$  is a feasible solution to  $LP_f$ . So the constraint is satisfied.

Finally, we consider any job  $j$  that had an edge removed in the cycle. Assume without loss of generality that  $(b_2, j)$  was removed from the graph. Since  $j$  is the root of the tree it is in (by line 11), all of its neighboring blocks are its children. Then, we have

$$\begin{aligned} \sum_i x_{i,j}^* \cdot f_j(i) &= \sum_{i \neq b_2} \lceil 2x_{i,j} \rceil \cdot f_j(i) \geq x_{b_1,j} \cdot f_j(b_1) + x_{b_2,j} \cdot f_j(b_2) + \sum_{i \neq b_1, b_2} 2x_{i,j} \cdot f_j(i) \\ &\geq \sum_i x_{i,j} \cdot f_j(i) \geq d_j. \end{aligned}$$

The second inequality comes as a consequence of line 10 and the fact  $(b_2, j)$  was removed from the graph, which implies that  $2x_{b_1,j} \cdot f_j(b_1) \geq x_{b_1,j} \cdot f_j(b_1) + x_{b_2,j} \cdot f_j(b_2)$ . So the constraint is satisfied in all cases. Thus  $(x^*, y^*)$  is a feasible integer solution to LP(1-4). ◀

► **Theorem 7.** *Algorithm 4 returns a  $\min\{(3 + \varepsilon)\text{OPT}, (2 + \varepsilon)\text{OPT} + |C|\}$  approximation to the CMS problem in polynomial time if the number of configurations is constant.*

**Proof.** We show that the algorithm returns a solution with the stated approximation factor. Consider the iteration where  $C^* = C^{\text{OPT}}$  where  $C^{\text{OPT}}$  is the set of configurations used by an optimal integer solution. The algorithm will iterate through potential counts  $m_\sigma$  for each  $\sigma$  in  $C^*$ , round and return a schedule the first time  $LP_f$  has a feasible solution; let  $m$  be the vector of how many configurations are used in this iteration. By Lemma 6, the solution returned is feasible.

We now bound the cost by first arguing that  $\sum_\sigma m_\sigma \leq (1 + \varepsilon)\text{OPT}$ . Observe that the  $y$  values in the optimal integer solution to LP(1-4) would yield a feasible solution to  $LP_f$  if they equalled the corresponding  $m$  values in  $LP_f$  (namely by setting the  $x$  variables in  $LP_f$  to the  $x$  values in the optimal integer solution to LP(1-4)). For each such  $y_i$  value, consider  $p_i$ , the first power of  $1 + \varepsilon$  that is at least  $y_i$ . Then, we have  $y_i \leq \lfloor p_i \rfloor \leq (1 + \varepsilon)y_i$ . By definition of  $L$ , we will set values for  $m_\sigma$  such that they are greater than and within a factor of  $(1 + \varepsilon)$  of the  $y$  values from the optimal integer solution. Thus they will be feasible, since they use at least as many of each configuration and  $\sum_\sigma m_\sigma \leq (1 + \varepsilon)\text{OPT}$ . Since we iterate through the  $m$  values in increasing order of  $\sum_\sigma m_\sigma$ , the first feasible solution will use at most this many configurations.

Now consider that the rounded solution  $y^*$  has  $\sum_\sigma y_\sigma^* \leq \sum_\sigma (2m_\sigma + 1) = 2\sum_\sigma m_\sigma + |C^{\text{OPT}}| \leq 2(1 + \varepsilon)\text{OPT} + |C^{\text{OPT}}|$ . Since the optimal integer solution uses at least 1 of each configuration in  $C^{\text{OPT}}$ , we have that  $\sum_\sigma y_\sigma^* \leq (3 + \varepsilon)\text{OPT}$  and also that  $\sum_\sigma y_\sigma^* \leq (2 + \varepsilon)\text{OPT} + |C|$ .

Finally, we prove that the runtime of algorithm 4 is polynomial if  $|C| = O(1)$ . The first for loop in the algorithm ranges over  $2^{|C|}$  values. The inner for loop ranges over  $(\log L)^{|C^*|}$  values. Remember that  $L = \sum_j d_j$ . But then  $L \leq n \cdot \max_j d_j$ . Thus the inner loop ranges over  $\leq (\log(n \cdot \max_j d_j))^{|C^*|} \leq (\log n + \log \max_j d_j)^{|C|}$  values. Since  $d_j$  is specified as a number, it is specified using  $\log d_j$  bits. Thus the inner loop runs a number of times polynomial in the input, except for the number of configurations. Lastly we analyze the body of the inner for loop. The size of the LP is polynomial in the size of the input, and thus constructing and solving it takes time polynomial in the size of the input. Constructing the graph takes time polynomial in the size of the LP, as does rounding using the graph. Thus overall the runtime of the algorithm is polynomial in the size of the input, except for it being exponential in the number of configurations. So if  $|C| = O(1)$ , the algorithm is polynomial. ◀

#### 4 CMS with a constant number of configurations of constant size

In this section, we consider CMS with  $n$  jobs, a set  $C$  of a constant number of configurations with the additional constraint that each configuration has at most a constant number  $b$  of blocks. Let  $k$  be the total number of block types. Since  $|C|$  and  $b$  are both constant,  $k \leq b|C|$  is a constant. In Section 4.2, we present our main result of this section, a PTAS for the problem. As a warmup, in Section 4.1, we present an optimal dynamic programming algorithm for the problem, which takes time  $(nbd_{\max})^{O(k+|C|)}$ ; this is pseudo-polynomial time for constant  $k$  and  $|C|$ .

##### 4.1 A pseudo-polynomial time algorithm

We present an optimal dynamic programming algorithm that takes time polynomial in  $n$  and the maximum demand. Recall that  $C$  denotes the set of configurations, and  $|C|$  is constant. Let  $N$  denote the total number of machines. Then, there are  $\binom{N+|C|-1}{|C|-1}$  ways of distributing the  $N$  machines among these configurations. Each way yields a specific number of blocks of each type. For given  $n_i$ ,  $1 \leq i \leq k$ , let  $S(j, n_1, n_2, \dots, n_k)$  be True if the demand of jobs 1 through  $j$  can be satisfied using  $n_i$  blocks of type  $i$ , for each  $i$ . Then, we have

$$S(j, n_1, n_2, \dots, n_k) = \bigvee_{m_i \leq n_i, \forall i} (S(j-1, n_1 - m_1, n_2 - m_2, \dots, n_k - m_k) \wedge T(j, m_1, m_2, \dots, m_k))$$

where  $T(j, m_1, m_2, \dots, m_k)$  is true if and only if  $d_j$  can be satisfied using  $m_i$  blocks of type  $i$ , for each  $i$ . Note that  $T(j, m_1, m_2, \dots, m_k)$  can be computed easily by inspecting the demand table of job  $j$  and  $d_j$ .

The algorithm computes  $S(j, n_1, n_2, \dots, n_k)$  for  $1 \leq j \leq n$ ,  $n_i \leq Nb$ ; the number of different tuples equals  $n(Nb)^k$ . The time taken to compute a given  $S(j, n_1, n_2, \dots, n_k)$ , given  $S(j-1, n_1 - m_1, n_2 - m_2, \dots, n_k - m_k)$  for all choices of  $m_i$ 's, is proportional to the number of different choices of  $m_i$ 's, which is bounded by  $\binom{N+|C|-1}{|C|-1}$ . We thus obtain that  $S$  can be computed in  $n(Nb)^{O(k+|C|)}$ . This computation, coupled with a binary search over possible values of  $N$ , yields the desired algorithm. Since  $N$  is bounded by  $n$  times the maximum demand, we obtain a pseudopolynomial time optimal algorithm if  $|C|$  and  $k$  are bounded.

##### 4.2 A polynomial-time approximation scheme

*Blocks and patterns.* Abusing notation slightly, we use  $f_j(\sigma)$  to denote the total demand of  $j$  satisfied if every block in configuration  $\sigma$  is assigned to  $j$ . We partition the set  $J$  of jobs into two groups: the *large* jobs  $L$  and *small* jobs  $S$ . A job  $j$  is small if there exists a configuration  $\sigma$  such that  $f_j(\sigma) \geq \varepsilon d_j$ ; otherwise,  $j$  is large.

Let  $\varepsilon > 0$  be a given constant parameter, and let  $\lambda = \varepsilon/(2b)$ . We define a *pattern*  $\pi$  to be a size  $k$  list of integers  $\pi_1$  through  $\pi_k$  that sum to no more than  $b/\lambda^2$ ;  $\pi_i$  denotes the number of blocks  $i$  in pattern  $\pi$ . Let  $W$  be the set of all possible patterns. So,  $|W| \leq (b/\lambda^2)^k$ . We assign each small job a *type*. Job  $j$  is of type  $t \in 2^W$  if each pattern  $\pi \in t$  is such that the demand of  $j$  is satisfied if  $j$  is allocated  $\pi_i$  blocks  $i$  for  $1 \leq i \leq k$ . So, the number of job types is at most  $2^{(b/\lambda^2)^k}$ . Define constant  $\gamma = 2^{(b/\lambda^2)^k}$ .

*The linear program.* We define a linear program PTAS-LP using the following notation. In PTAS-LP,  $\sigma$  ranges over all configurations in  $C$ ,  $p \in \{1, \dots, k\}$  ranges over types of blocks,  $x_{j,p}$  is the number of  $p$ -blocks dedicated to processing a large job  $j$ ,  $y_\sigma$  is the number of

machines we use with configuration  $\sigma$ ,  $\sigma_p$  is the number of  $p$ -blocks in  $\sigma$ ,  $z_{t,\pi}$  is the number of small jobs of type  $t$  that are distributed according to pattern  $\pi$ , and  $n_t$  is the number of small jobs of type  $t$ . Recall that  $\pi_p$  is the  $p$ th entry of  $\pi$ . PTAS-LP minimizes  $\sum_{\sigma \in C} y_\sigma$  subject to the following constraints

$$\sum_{j \in L} x_{i,j} + \sum_{t \in 2^W} \sum_{\pi \in W} (z_{t,\pi} \cdot \pi_i) \leq \sum_{\sigma} y_\sigma \cdot \sigma_i \quad i \in [k] \quad (5)$$

$$\sum_{i \in [k]} f_j(p) \cdot x_{i,j} \geq d_j \quad j \in L \quad (6)$$

$$\sum_{\pi} z_{t,\pi} \geq n_t \quad t \in 2^W \quad (7)$$

$$x_{i,j}, y_\sigma, z_{t,\pi} \geq 0 \quad j \in L, i \in [k], \sigma, t \in 2^W, \pi \in W \quad (8)$$

**Constraints.** Constraint 5 guarantees that the number of blocks  $i$  that are used to execute jobs is at most the number of available blocks  $i$ . Constraint 6 ensures that each large job is fully executed, and constraint 7 guarantees that each small job is fully executed. Constraint 8 ensures non-negativity.

Lemma 8 proves that it is sufficient to consider schedules in which small jobs are executed by a bounded number of blocks. Lemma 9 uses Lemma 8 and shows PTAS-LP is an approximate relaxation for the problem.

► **Lemma 8.** *For any schedule with  $m$  machines, there exists a schedule with  $m(1 + b\lambda)$  machines in which each small job is executed by at most  $b/\lambda^2$  blocks.*

**Proof.** Consider any placement  $P$  that uses  $m$  machines. Suppose a small job  $j$  is in more than  $b/\varepsilon^2$  blocks in  $P$ . Since each configuration is of size at most  $b$ , it follows that  $j$  is placed in at least  $1/\lambda^2$  machines. Since  $j$  is small, there exists a configuration  $\sigma$  such that  $f_j(\sigma) \geq \lambda d_j$ . We remove  $j$  from each machine to which it is assigned in  $P$  and place it in  $1/\lambda$  additional machines, each with configuration  $\sigma$ , guaranteeing that the demand of  $j$  is satisfied. Since each machine can hold at most  $k$  small jobs, this modification of  $P$  results in the increase in the number of machines by a factor of at most  $(1 + b\lambda)$ , yielding the desired claim. ◀

► **Lemma 9.** *The value of PTAS-LP is at most  $(1 + b\lambda)\text{OPT}$ .*

**Proof.** Let  $A$  be an optimal placement of the jobs on  $m$  machines. Using Lemma 8, we first compute a new placement  $B$  using at most  $m(1 + b\lambda)$  machines in which each small job is placed in at most  $1/\lambda^2$  machines.

We now define variable assignments so that the value of PTAS-LP is no more than  $(1 + b\lambda)m$ . For each large job  $j$  and each block  $i$ , set  $x_{i,j}$  to be the number of blocks  $i$  on which  $B$  executes  $j$ . For each small job type  $t$  and each pattern  $\pi$ , set  $z_{t,\pi}$  to be the number of small jobs that are executed in pattern  $\pi$  according to  $B$ . Note that since each small job is placed in at most  $1/\lambda^2$  machines, and hence at most  $b/\lambda^2$  blocks, the placement of each small job follows one of the patterns in  $W$ . Set  $y_\sigma$  equal to the number of machines with configuration  $\sigma$  according to  $A$ .

It is easy to see that constraints (6 - 8) are satisfied. To see that constraint 5 is satisfied, observe that each machine used by  $B$  either has some block executing a large job (in which case it contributes toward the first term of 5) or it has some block executing a small job (in which case it contributes toward the second term). Therefore, the left hand side of 5 counts the total number of blocks needed to complete all the jobs, while the right hand side computes the total number of blocks supplied by the machines. ◀

■ **Algorithm 5** Schedule for  $O(1)$  configurations of  $O(1)$  size.

---

**Input:**  $(C, J, k)$

- 1 Solve PTAS-LP; let  $(x, y, z)$  be the solution computed.
- 2 **if**  $n \leq b(|C| + \gamma)/\lambda$  **then**
- 3    └ Compute and return an optimal solution using enumeration
- 4 **foreach** large job  $j$  and block  $i$  **do**
- 5    └  $\hat{x}_{i,j} = \lceil x_{i,j} \rceil$ ; Assign  $\lceil x_{i,j} \rceil$  blocks  $i$  to job  $j$
- 6 **foreach** job type  $t$  and pattern  $\pi$  **do**
- 7    └ Assign blocks per pattern  $\pi$  to each job in  $\lceil z_{t,\pi} \rceil$  small jobs of type  $t$
- 8 **foreach** configuration  $\sigma$  **do**
- 9    └ Use  $\lceil y_\sigma \rceil$  machines with configuration  $\sigma$

---

► **Theorem 10.** *Algorithm 5 returns a  $(1+\varepsilon)$  approximation to the CMS problem in polynomial time if the number of configurations is constant and they are of constant size.*

**Proof.** First, if  $n \leq b(|C| + \gamma)/\lambda$ , then the algorithm returns an optimal solution. Otherwise, since each machine has at most  $b$  blocks, we obtain that  $\text{OPT} \geq (|C| + \gamma)/\lambda$ . We will show that the number of machines used is at most  $(1 + b\lambda)\text{OPT} + \lambda^2 b\text{OPT} + |C| + \gamma$ , which is at most  $(1 + 2b\lambda)\text{OPT} = (1 + \varepsilon)\text{OPT}$ .

Rounding up the  $x$  variables increases the number of blocks by at most the number of large jobs times the number of block types. Since each large job requires at least  $1/\lambda^2$  machines, this increase in the number of blocks is at most  $\lambda^2 b\text{OPT}$ . Rounding up the  $z$  variables adds at most  $1/\lambda^2$  blocks per small job type assigned to a given pattern, increasing the number of blocks by at most  $\gamma$ . Rounding up the  $y$  variables increases the number of machines by  $|C|$ . Taken together with the above increase in the number of blocks, each of which requires at most one machine, the total increase is bounded by  $\lambda^2 b\text{OPT} + \gamma + |C|$ . By Lemma 9, the LP optimal is at most  $(1 + b\lambda)\text{OPT}$ , yielding the desired claim.

The linear program PTAS-LP has at most  $nk + |C| + \gamma \log \gamma$  variables and  $k + n + \gamma$  linear constraints (other than the non-negativity ones), and can be solved in polynomial time. The enumeration for  $n \leq b(|C| + \gamma)/\lambda$  is constant time, while the rest of the algorithm is linear in the number of variables. The hidden constant, however, is doubly exponential in  $|C|$  and the configuration size bound  $b$ , and exponential in  $1/\varepsilon$ . ◀

## 5 CMS with $O(1)$ number jobs and block types, and all configurations up to a given size

In this section, we consider models for which we are able to obtain optimal solutions in polynomial time. For models with a constant number of jobs, blocks, and configurations, the linear program (1-4) has constant size and so can be solved as an integer program. Many models, however, do not fit this framework and require more sophisticated methods. Consider a setting where the number  $n$  of jobs is constant, as well as the number  $k$  of block types. We also assume that there exists an integer  $K$  such that the set of available configurations  $C$  includes all configurations up to size  $K$ . For example, with  $k = 2$  block types  $\{1, 2\}$  and  $K = 3$ , the set of available configurations  $C = \{\{1, 1, 1\}, \{1, 1, 2\}, \{1, 2, 2\}, \{2, 2, 2\}\}$ . In this section, we show that a class of problems, including this model, is polynomial time solvable using methods developed by Goemans and Rothvoss [5] to optimally solve bin packing with a constant number of job types.

Consistent with our initial description of the CMS model, we assume all configurations are given as input. However, the input can be reduced exponentially if configurations are given as a single parameter  $K$  indicating the size of all valid configurations. The algorithm described in this section is polynomial time in either case – that is, the algorithm’s runtime is polynomial in  $\log K$  and  $\log \max_{i,j} \{d_j/f_j(i)\}$  (assuming  $n$  and  $k$  are constant). We note that there is an easy reduction from bin packing with constant job types to this problem (when configurations are not listed individually).

Let  $P$  be the set of vectors:

$$\begin{array}{l} \left[ \begin{array}{c} u_{1,1} \\ \vdots \\ u_{k,n} \\ v_1 \\ \vdots \\ v_n \\ 1 \end{array} \right] \text{ subject to} \\ \sum_{j=1}^n \sum_{i=1}^b u_{i,j} \leq K \quad (9) \\ v_j = \sum_{i=1}^k u_{i,j} \cdot f_j(i) \quad j \in J \quad (10) \\ u_{i,j} \geq 0 \quad i \in [k], j \in J \quad (11) \\ v_j \geq 0 \quad j \in J \quad (12) \end{array}$$

Let  $Q_H$  be the set of vectors

$$\begin{array}{l} \left[ \begin{array}{c} z_{1,1} \\ \vdots \\ z_{b,n} \\ w_1 \\ \vdots \\ w_n \\ h \end{array} \right] \text{ subject to,} \\ 0 \leq h \leq H \quad (13) \\ w_j \geq d_j \quad j \in J \quad (14) \\ z_{i,j} \geq 0 \quad i \in [k], j \in J \quad (15) \end{array}$$

► **Lemma 11** (Theorem 2.2 in [5]). *Let  $N = nk + n + 1$ . Given rational polyhedra  $P, Q \in \mathbb{R}^N$  where  $P$  is bounded, one can find a vector  $y \in \text{int.cone}(P \cap \mathbb{Z}^N) \cap Q$  and a vector  $\lambda \in \mathbb{Z}_{\geq 0}^{|P \cap \mathbb{Z}^N|}$  such that  $y = \sum_{x \in P \cap \mathbb{Z}^N} \lambda_x x$  in time  $\text{enc}(P)^{2^{O(N)}} \cdot \text{enc}(Q)$ , or decide that no such  $y$  exists. Moreover, the support of  $\lambda$  is bounded by  $2^{2N+1}$ .*

► **Lemma 12.** *Given  $P$  and  $Q_H$ , the algorithm of Lemma 11 returns a vector  $\lambda$  iff there exists a solution using at most  $H$  machines. Moreover, a returned vector  $\lambda$  provides a solution that uses at most  $H$  machines.*

**Proof.** As in the lemma, let  $N = nk + n + 1$ . We first show that, if there exists a valid solution using at most  $H$  machines then  $\text{int.cone}(P \cap \mathbb{Z}^N) \cap Q_H \neq \emptyset$ . Suppose there exists a valid solution to the problem using at most  $H$  machines. We argue that each machine  $\mu$  corresponds to an integer valued vector in  $P$ . Consider an arbitrary machine  $\mu$  in the given solution. Recall that  $\mu(i, j)$  is the number of blocks of type  $i$  on which  $\mu$  executes job  $j$ . This implies that  $\sum_j \sum_i \mu(i, j) \leq K$ . Setting  $v_j = \sum_i \mu(i, j) \cdot f_j(i)$  ensures that the vector  $(v_1 \dots v_n \ 1 \ u_{1,1} \dots u_{b,n}) \in P$ . Let  $p_\mu$  be the vector in  $P$  corresponding to machine  $\mu$ . Let  $\lambda = (\lambda_\mu)$ , where  $\lambda_\mu$  is the multiplicity of machine  $\mu$  in the given solution. Then  $\sum_\mu \lambda_\mu \cdot p_\mu$  yields integer vector  $q = (z_{1,1} \dots z_{b,n} \ w_1 \dots w_n \ h)$ . Since every job is fully executed, we can infer that  $w_j \geq d_j$ . Since the solution uses at most  $H$  machines, we can infer that  $0 \leq h \leq H$ . Therefore,  $q \in Q$ , which implies that  $\text{int.cone}(P \cap \mathbb{Z}^N) \cap Q \neq \emptyset$ .

We now show that, if  $\text{int.cone}(P \cap \mathbb{Z}^N) \cap Q_H \neq \emptyset$ , then there exists a solution to the problem using at most  $H$  machines. Suppose there exists a vector  $q \in \text{int.cone}(P \cap \mathbb{Z}^N) \cap Q_H$ . Define  $\lambda = (\lambda_x)_{x \in P \cap \mathbb{Z}^N}$  such that  $\sum_{x \in P \cap \mathbb{Z}^N} x \cdot \lambda_x = q$ . As above, we treat each element  $x \in P \cap \mathbb{Z}^N$  as corresponding to a machine, and  $\lambda_x$  as the multiplicity of that machine in our final schedule. Similar reasoning shows that  $\lambda$  yields a valid solution, which completes the proof of the lemma. ◀

► **Theorem 13.** *Given an instance of CMS with  $O(1)$  jobs and  $O(1)$  block types, and where all configurations of size at most  $K$  are available, there exists an algorithm that computes an optimal solution in  $\text{poly}(\log K, \log \max_{i,j} \{d_j/f_j(i)\})$  time.*

**Proof.** The maximum number of machines needed in any solution is  $m = n \cdot \max_{i,j} \{d_j/f_j(i)\}$ . We binary search in the range  $[0, m]$  to find the minimum integer value of  $H$  for which the algorithm of Lemma 11, given  $P$  and  $Q_H$ , returns a vector  $\lambda$ . By Lemma 12 this provides an optimal solution. ◀

We have shown the existence of an algorithm that computes an optimal solution to CMS with  $O(1)$  number of jobs and block types, and with all configurations up to a given size. However, the Goemans-Rothvoss framework generalizes to a constant number of polytopes (Theorem 6.2 in [5]) which entails the following stronger claim.

► **Lemma 14.** *Suppose an instance of CMS has a constant number of jobs and blocks, and the set of all configurations can be specified using a set  $T$  of rational polytopes with  $|T| = O(1)$ . Then there exists an algorithm that computes an optimal solution in time polynomial in  $\log K$  and  $\log \max_{i,j} \{d_j/f_j(i)\}$ .*

To prove Lemma 14, we use the following result from Goemans and Rothvoss.

► **Lemma 15** (Theorem 6.2 in [5]). *Given rational polytopes  $\{P_t \subseteq \mathbb{R}^{nk+n+1} : t \in T\}$  and rational polyhedron  $Q \subseteq \mathbb{R}^{n+1}$ , define  $X_t := \{x \in \mathbb{Z}^{n+1} : \exists y \in \mathbb{Z}^{nb}, (x, y) \in P_t\}$ . Then there is an algorithm that decides correctly whether  $\text{int.cone}(\bigcup_{t \in T} X_t) \cap Q \neq \emptyset$  in time  $(\text{enc}(P))^{2^{O(nk+n+1+|T|)}} \cdot \text{enc}(Q)^{O(1)}$ . In the affirmative case, the algorithm provides a vector  $\lambda = (\lambda_{t,x})_{t \in T, x \in X}$  with  $\lambda_{t,x} \in \mathbb{Z}_{\geq 0}$  and  $(\sum_{t \in T} \sum_{x \in X} \lambda_x \cdot x) \in Q$ . Moreover, the size of the support of  $\lambda$  is bounded by  $2^{2(nk+n+1+|T|)+1}$ .*

**Proof of Lemma 14.** For each  $t \in T$ , we specify  $P_t$  as an  $nk + n + 1$  length vector as above, except the polytope is defined by arbitrary rational constraints specific to  $P_t$ .  $Q_H$  is defined as above. As in Lemma 12, we argue the following claim: given  $\{P_t\}_{t \in T}$  and  $Q_H$ , the algorithm of Lemma 15 returns a vector  $\lambda$  iff there exists a solution using at most  $H$  machines. Moreover, a returned vector  $\lambda$  provides a solution that uses at most  $H$  machines. The reasoning is similar to that used in Lemma 12.

Let  $X_t$  be defined as in the lemma, for all  $t \in T$ . If there is a valid solution then for each machine  $\mu$  there is some  $t$  such that  $\mu$  can be represented as an element  $p$  of  $P_t$ . Also, we define  $\lambda = (\lambda_{t,p})_{t \in T, p \in X}$  such that  $\lambda_{t,p}$  provides the multiplicity of the machine corresponding to  $p$  in the solution. The fact that we begin with a valid solution on at most  $H$  machines ensures that  $\sum_{t \in T} \sum_{x \in X} x \cdot \lambda_{t,x} \in Q_H$ . In the other direction, we show that if the algorithm returns a vector  $\lambda$ , then  $\lambda$  provides a solution to the problem. As above, we can interpret  $\lambda_{t,x}$  as providing the multiplicity of the machine corresponding to  $x \in X$  in our solution – since there exists  $y$  such that  $(x, y) \in P_t$ , we can infer that  $x$  is a valid machine. By the constraints on  $Q_H$ , we can infer that the derived solution completes all jobs and uses at most  $H$  machines. This proves the lemma. ◀

---

## References

- 1 Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 31–40, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1132516.1132522.
- 2 Deeparnab Chakrabarty, Julia Chuzhoy, and Sanjeev Khanna. On allocating goods to maximize fairness. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 107–116, 2009. doi:10.1109/FOCS.2009.51.
- 3 S. W. Cheng and Y. Mao. Restricted max-min allocation: Integrality gap and approximation algorithm. *Algorithmica*, 84:1835–1874, 2022.



- 4 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591884.
- 5 Michel X. Goemans and Thomas Rothvoss. Polynomiality for bin packing with a constant number of item types. *J. ACM*, 67(6), November 2020. doi:10.1145/3421750.
- 6 Qiang-Sheng Hua, Amy Wang, Dongxiao Yu, and Francis Lau. Dynamic programming based algorithms for set multicover and multiset multicover problem. *Theor. Comput. Sci.*, 411:2467–2474, June 2010. doi:10.1016/j.tcs.2010.02.016.
- 7 Zhihao Jiang and Haoyu Zhao. An fptas for stochastic unbounded min-knapsack problem. In Yijia Chen, Xiaotie Deng, and Mei Lu, editors, *Frontiers in Algorithmics*, pages 121–132, Cham, 2019. Springer International Publishing.
- 8 B. Korte and J. Vygen. *Bin-Packing*, pages 426–441. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. doi:10.1007/3-540-29297-7\_18.
- 9 Jan Karel Lenstra, David B. Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 217–224, 1987. doi:10.1109/SFCS.1987.8.
- 10 Baolin Li, Tirthak Patel, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Miso: exploiting multi-instance gpu capability on multi-tenant gpu clusters. In *Proceedings of the 13th Symposium on Cloud Computing*, pages 173–189, 2022.
- 11 Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Clover: Toward sustainable ai with carbon-aware machine learning inference service. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2023.
- 12 R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, EC '04, pages 125–131, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/988772.988792.
- 13 NVIDIA Multi-Instance GPU User Guide. [https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA\\_MIG\\_User\\_Guide.pdf](https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf), 2024.
- 14 Deepak Narayanan, Fiodar Kazhamiaka, Firas Abuzaid, Peter Kraft, Akshay Agrawal, Srikanth Kandula, Stephen Boyd, and Matei Zaharia. Solving large-scale granular resource allocation problems efficiently with pop. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 521–537, 2021.
- 15 Sridhar Rajagopalan and Vijay V. Vazirani. Primal-dual rnc approximation algorithms for set cover and covering integer programs. *SIAM J. Comput.*, 28:525–540, 1999. URL: <https://api.semanticscholar.org/CorpusID:36747871>.
- 16 Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 322–337, 2019.
- 17 Cheng Tan, Zhichao Li, Jian Zhang, Yu Cao, Sikai Qi, Zherui Liu, Yibo Zhu, and Chuanxiong Guo. Serving DNN models with multi-instance GPUs: A case of the reconfigurable machine scheduling problem, 2021. arxiv:2109.11067. arXiv:2109.11067.
- 18 Vijay V. Vazirani. *Approximation Algorithms*. Springer Publishing Company, Incorporated, 2010.
- 19 Wencong Xiao, Shiru Ren, Yong Li, Yang Zhang, Pengyang Hou, Zhi Li, Yihui Feng, Wei Lin, and Yangqing Jia. {AntMan}: Dynamic scaling on {GPU} clusters for deep learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 533–548, 2020.

## A Constructing Schedules of Polynomial Size

For any schedule  $S$  with  $m$  machine instances, there exists a multiset of configurations  $T$  used in  $S$  and, for each job  $j$ , a multiset of blocks  $T_j$  used in  $S$ . Note both  $\text{enc}(T)$  and  $\text{enc}(T_j)$ , for every  $j$ , are polynomial in  $|C|$ ,  $|J|$ , and  $k$ , where  $\text{enc}(A)$  denotes the binary encoding length of a multiset  $A$ .

► **Lemma 16.** *Given a multiset of configurations  $T$  and, for each job  $j$ , a multiset of blocks  $T_j$ , we can output a schedule  $M$  such that (i) the number of block instances across all machines instances in  $M$  is at least the number of block instances in  $T$ , (ii) the multiset of blocks assigned to each job  $j$  across all machine instances in  $M$  is identical to  $T_j$ , and (iii) the description of  $M$  has length polynomial in  $\text{unique}(T) + \sum_j \text{unique}(T_j)$ , where  $\text{unique}(A)$  denotes the number of distinct elements in multiset  $A$ .*

**Proof.** We construct  $M$  via Algorithm 6. We show that the number of different machines in the schedule produced by Algorithm 6 is polynomial in  $|T| + \sum_j |T_j|$ . A new machine is constructed in each iteration of the while-loop. In a single iteration of the while-loop, machines are allocated until condition (i) or condition (ii) holds. By the line 10 and 11 updates, the number of times these conditions can be met is at most  $\text{unique}(T) + \sum_j \text{unique}(T_j)$ . This proves the claim. ◀

■ **Algorithm 6** Multiset-to-Machines.

---

**Input:**  $(T, T_1, \dots, T_j)$

- 1 for all  $\sigma : s_\sigma \leftarrow$  the number of occurrences of  $\sigma$  in  $T$
- 2 for all  $i, j : t_{i,j} \leftarrow$  the number of occurrences of  $i$  in  $T_j$
- 3 **while**  $s_\sigma + \sum_{i,j} t_{i,j} > 0$  **do**
- 4     choose any  $\sigma \in T$  such that  $s_\sigma > 0$
- 5     construct a new machine  $\mu$  with configuration  $\sigma$
- 6     **foreach**  $i \in [k]$  **do**
- 7         assign  $\sigma_i$  jobs to block  $i$  in  $\sigma$ , ensuring that for every job  $j$ ,  $\mu(i, j) \leq t_{i,j}$
- 8     allocate  $a$  instances of  $\mu$ , where  $a$  is the minimum value such that  
        either (i)  $a = s_\sigma$ , or (ii) for some  $i, j$ ,  $a \cdot \mu(i, j) = t_{i,j}$
- 9     for all  $i, j : t_{i,j} \leftarrow t_{i,j} - a \cdot \mu(i, j)$
- 10     $s_\sigma \leftarrow s_\sigma - a$

---

# On Instance-Optimal Algorithms for a Generalization of Nuts and Bolts and Generalized Sorting

Mayank Goswami  

Queens College CUNY, Flushing, New York, USA

Riko Jacob  

IT University of Copenhagen, København S, Denmark

---

## Abstract

We generalize the classical nuts and bolts problem to a setting where the input is a collection of  $n$  nuts and  $m$  bolts, and there is no promise of any matching pairs. It is not allowed to compare a nut directly with a nut or a bolt directly with a bolt, and the goal is to perform the fewest nut-bolt comparisons to discover the partial order between the nuts and bolts. We term this problem *bipartite sorting*.

We show that instances of bipartite sorting of the same size exhibit a wide range of complexity, and propose to perform a fine-grained analysis for this problem. We rule out straightforward notions of instance-optimality as being too stringent, and adopt a *neighborhood-based* definition. Our definition may be of independent interest as a unifying lens for instance-optimal algorithms for other static problems existing in literature. This includes problems like sorting (Estivill-Castro and Woods, ACM Comput. Surv. 1992), convex hull (Afshani, Barbay and Chan, JACM 2017), adaptive joins (Demaine, López-Ortiz and Munro, SODA 2000), and the recent concept of universal optimality for graphs (Haeupler, Hladík, Rozhoň, Tarjan and Tětek, 2023).

As our main result on bipartite sorting, we give a randomized algorithm that is within a factor of  $O(\log^3(n+m))$  of being instance-optimal w.h.p., with respect to the neighborhood-based definition.

As our second contribution, we generalize bipartite sorting to DAG sorting, when the underlying DAG is not necessarily bipartite. As an unexpected consequence of a simple algorithm for DAG sorting, we rule out a potential lower bound on the widely-studied problem of *sorting with priced information*, posed by (Charikar, Fagin, Guruswami, Kleinberg, Raghavan and Sahai, STOC 2000). In this problem, comparing keys  $i$  and  $j$  has a known cost  $c_{ij} \in \mathbb{R}^+ \cup \{\infty\}$ , and the goal is to sort the keys in an instance-optimal way, by keeping the total cost of an algorithm as close as possible to  $\sum_{i=1}^{n-1} c_{x(i)x(i+1)}$ . Here  $x(1) < \dots < x(n)$  is the sorted order. While several special cases of cost functions have received a lot of attention in the community, no progress on the general version with arbitrary costs has been reported so far. One reason for this lack of progress seems to be a widely-cited  $\Omega(n)$  lower bound on the competitive ratio for *finding the maximum*. This  $\Omega(n)$  lower bound by (Gupta and Kumar, FOCS 2000) uses costs in  $\{0, 1, n, \infty\}$ , and although not extended to sorting, this barrier seems to have stalled any progress on the general cost case.

We rule out such a potential lower bound by showing the existence of an algorithm with a  $\tilde{O}(n^{3/4})$  competitive ratio for the  $\{0, 1, n, \infty\}$  cost version. This generalizes the setting of *generalized sorting* proposed by (Huang, Kannan and Khanna, FOCS 2011), where the costs are either 1 or infinity, and the cost of the cheapest proof is always  $n-1$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Abstract machines; Theory of computation  $\rightarrow$  Sorting and searching

**Keywords and phrases** Sorting, Priced Information, Instance Optimality, Nuts and Bolts

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.23

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2211.04601> [15]



© Mayank Goswami and Riko Jacob;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 23; pp. 23:1–23:23



Leibniz International Proceedings in Informatics

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

**Funding** *Mayank Goswami*: Supported by NSF grant CCF-1910873.

*Riko Jacob*: Part of this work done during the second Hawaiian workshop on parallel algorithms and data structures, University of Hawaii at Manoa, Hawaii, USA, NSF Grant CCF-1930579.

## 1 Introduction

The classic nuts-and-bolts problem, originally mentioned as an exercise in [27], asks: a disorganized carpenter has  $n$  nuts and  $n$  bolts, and there is (perfect) matching bolt for every nut. The only allowed comparison is between a nut and a bolt, and the result of such a comparison is either  $<$ ,  $=$  or  $>$ . The goal is to find the matching without comparing two nuts or two bolts to each other. A simple Quicksort type algorithm can be shown to solve this problem in optimal  $O(n \log n)$  comparisons with high probability: Pick a random nut, compare to all bolts, find the matching bolt, and compare that bolt to all nuts. The problem is now partitioned into two subproblems with the match at the boundary; recurse. In a later work [3], Alon, Blum, Fiat, Kannan, Naor and Ostrovsky developed a deterministic Quicksort-type algorithm that uses expanders and performs  $O(n \text{ polylog } n)$  comparisons. This was later improved to an optimal  $O(n \log n)$  comparisons by Komlós, Ma, Szémeredi [24], by performing substantial modifications on AKS sorting.

Our starting point is a remark by Komlós, Ma and Szémeredi: “In particular, the fact that we can sort the nuts and bolts at all relies on the fact that there is a match between them.” Indeed, if there is no matching (all comparisons come out  $<$  or  $>$ ), one realizes that the above randomized Quicksort based algorithm fails, as there is no partitioning into subproblems. The only case where sorting without a matching is possible is when nuts and bolts alternate in the final sorted order. Call this the perfectly interleaved case. Komlós, Ma and Szémeredi observed (in a private communication with Aumann) that their AKS sorting-based algorithm sorts the nuts and bolts using  $O(n \log n)$  comparisons in this setting<sup>1</sup>. However, a general instance may not be perfectly interleaved, and this setting was left open.

**Generalized Nuts and Bolts.** We focus on the problem alluded to by Komlós, Ma, and Szémeredi: what if the carpenter is completely disorganized, and has an unequal collection of nuts and bolts, without any matching pairs<sup>2</sup>? That is, assume the carpenter has a set  $R$  of  $n$  nuts and a set  $B$  of  $m$  bolts, and is only allowed to compare a nut to a bolt, and the result is either  $<$ , or  $>$ . Unless  $m = n$  and we are in the perfectly interleaved case, sorting  $R \cup B$  is not possible: there could be two (or more) nuts (resp. bolts) that compare the same way to all the bolts (resp. nuts). A natural goal for the carpenter now is to “sort as much as you can”, i.e., partition the set of nuts  $R$  into subsets  $R_1, R_2, \dots$  such that for any  $r, r' \in R_i$  and any  $b \in B$ ,  $r$  and  $r'$  have the same order with  $b$  (either both are smaller, or both are larger), and vice-versa (see figure 1). We term this generalization of nuts and bolts as *bipartite sorting*: given the complete bipartite graph  $G = (R \cup B, E)$ , the goal is to discover the orientation<sup>3</sup> on all the edges in  $E$  by querying as few of them as possible.

**The need for fine-grained analysis.** If all nuts are smaller than all bolts, it is clear that  $nm$  comparisons are needed by any algorithm, and obviously this number of comparisons suffices for any instance. Recall that the perfectly interleaved case can be solved (in fact

<sup>1</sup> For a simple randomized algorithm that does the same, see Appendix A.1 of full version [15].

<sup>2</sup> It can be observed that having some matchings in the input only makes it easier to solve.

<sup>3</sup> An edge  $e = (u, v)$  in  $G$  is oriented as  $u\bar{v}$  if  $u < v$  in the underlying DAG.



■ **Figure 1** An example output to an instance of the bipartite sorting problem. Continuous runs of incomparable nuts and bolts are called “stripes”.

completely sorted) much faster with only  $O(n \log n)$  comparisons. Assuming for simplicity that  $m = \Theta(n)$ , the inherent complexities of instances range in  $[\tilde{O}(n), \Omega(n^2)]$ . Is there a way to define instance-optimality for bipartite sorting that captures its variety of underlying instances, and is there a good instance-optimal algorithm?

It is worthwhile to imagine the DAGs for the above two instances: if all nuts are smaller than all bolts, the DAG  $\vec{G} = (V = R \cup B, \vec{E})$  will have all edges oriented from  $R$  to  $B$ , whereas the DAG in the perfectly interleaved case will be the transitive closure of the oriented edges in the *directed Hamiltonian path corresponding to the sorted order of the nuts and the bolts*. A *transitive reduction* of a DAG is the fewest number of oriented edges that by transitivity imply all other orientations. For the first instance, the transitive reduction has size  $nm$ , whereas for the second case, it has size  $n - 1$ , and therefore if we ignore  $\log^{O(1)} n$  factors, for both instances, the sizes of their transitive reductions matches their complexity. This immediately suggests a parametrization similar to an output-sensitive setting: the number of comparisons of a good algorithm should be close to (say, within log factors of) the size of the transitive reduction of the underlying DAG.

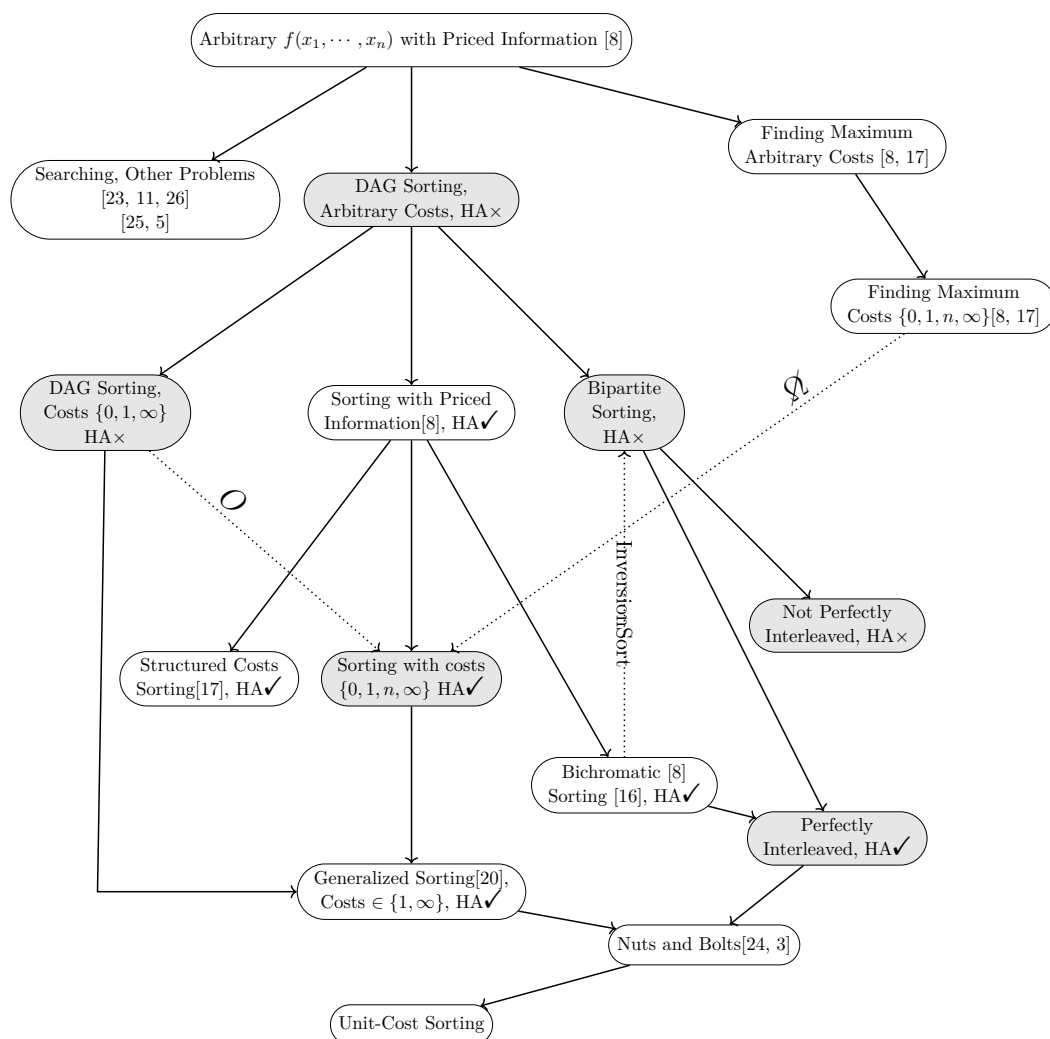
However, the following instance dashes all hopes of an algorithm that performs only roughly as many queries as there are edges in the transitive reduction:  $n - 1$  nuts are all smaller than a special bolt  $b$ , which is smaller than a special nut  $r$ , which is smaller than all other  $n - 1$  bolts. We call this the “*one-inversion*” instance. Even though the transitive reduction here has size  $2n - 1$ , any algorithm must perform  $n^2$  comparisons to find the hidden special pair  $(r, b)$ . Thus the gap between the transitive reduction and the inherent complexity of this instance is  $\Omega(n)$ , which is as large as the gap between the complexity of any two instances. Observe that the DAG for this instance is in some sense just “one-flip-away” from the all-nuts-smaller-than-all-bolts DAG, a phenomenon that will be important later. Given that the transitive reduction fails to capture the instances, we ask:

*Is there another meaningful way to define instance-optimality for bipartite sorting that captures its variety of underlying instances, and is there a good instance-optimal algorithm?*

## 2 Our Results and Technical Overview

In this paper, we answer the above question in the affirmative, and in the process of doing so, make unexpected progress on a widely-studied problem called generalized sorting (Huang, Kannan and Khanna [20]), which in turn is a special case of the sorting with priced information problem introduced by Charikar et al. [8]. We explain this connection first before stating our results.

**DAG Sorting.** One can generalize bipartite sorting to DAG sorting where the set of allowed comparisons is represented by an arbitrary (not necessarily bipartite) graph  $G = (V, E)$ . The goal is still to discover all orientations in the underlying DAG  $\vec{G} = (V, \vec{E})$  or equivalently, its transitive reduction, by querying as few edges of  $G$  as possible, where a query of an edge  $(u, v) \in E$  returns  $<$  or  $>$ . Edges not present in  $E$  cannot be queried.



■ **Figure 2** The landscape of sorting with priced information. Solid arrows go from a problem to its special case. HA✓ indicates that the Hamiltonian path is assumed to exist and HA× indicates that a Hamiltonian path may not exist. Problems shaded in gray are introduced or studied in this paper for the first time. Dotted arrows highlight our results, arrows with *O* show algorithms carrying over from one problem to another, and *Q* show lower bounds not carrying over.

It turns out that under the promise that  $\vec{G} = (V, \vec{E})$  has a directed Hamiltonian path, DAG sorting is exactly equivalent to generalized sorting. Thus both bipartite sorting<sup>4</sup> and generalized sorting can be viewed as special cases of DAG sorting. In fact, one can go a step further and assign non-negative costs to the edges in  $E$  in DAG sorting, and now ask for algorithms that find the transitive reduction with the cheapest cost. If the underlying DAG has a Hamiltonian path, this is exactly the problem of sorting with priced information.

<sup>4</sup> And by transitivity, sorting nuts and bolts when perfectly interleaved, matching nuts and bolts, and classical sorting.

## 2.1 Bipartite Sorting

As our main result, we define a meaningful notion of *instance optimality* for bipartite sorting, and give an algorithm that is instance optimal up to a factor of  $O(\log^3(n))$ . We found this (definition, algorithm) result surprising because it shows that in some sense, the “one-flip” phenomenon (recall the “one-inversion” instance) mentioned in the introduction is the *only* obstruction to achieving (almost) instance optimality. We believe our definition may be of independent interest as it unifies several previously-studied notions of instance optimality.

### 2.1.1 Defining Instance-Optimality

The transitive reduction of  $\vec{G}$ , denoted as  $\vec{T}$ , can be thought of as the “cheapest proof” of the underlying DAG being  $\vec{G}$ . Such a comparison to the cheapest proof has historically been very useful in defining instance optimality. Indeed, for the problems of generalized sorting and sorting with priced information, when the transitive reduction equals the directed Hamiltonian path, the cost of this directed Hamiltonian path is the measure of instance-optimality (and the factor by which an algorithm exceeds it is called *competitive ratio* in the original work by Charikar et al.[8]). The cheapest proof appears again in the work by Demaine, López-Ortiz and Munro [10], who study the problems of comparison-based set unions, intersections and differences. However in our setting, as the one-inversion instance shows, the size of the transitive reduction  $|\vec{T}|$  is *too strong a benchmark* to compare the performance of an algorithm to.

An important work on instance optimality deviating from a comparison to cheapest proof is by [Afshani, Barbay and Chan,[1]] which studies instance optimal algorithms for the convex hull problem. This was also adopted by [Cardinal, Dallant and Iacono [6]], who studied bichromatic rectangular visibility. When  $x$  is an input sequence of points, [1] and [6] define  $OPT(x)$  to be the runtime of an algorithm that is *order-oblivious*, i.e., OPT’s code can depend on the set of elements in  $x$ , but not on their order. The fact that *some* restriction on OPT is needed (at least for static problems) follows because comparing the runtime of an algorithm  $A$  on an input  $x$  to an algorithm (OPT) that is “tailor-made” for an input  $x$  is too strong: if OPT’s code can depend on  $x$ ,  $OPT(x)$  can be very small compared to  $A(x)$ . In the convex hull problem  $OPT(x)$  would just be  $O(n)$  as OPT only needs to read the input<sup>5</sup>, and for DAG/bipartite sorting if OPT knows the underlying DAG  $\vec{G}$  then (since we only count query complexity) OPT equals zero!

We posit a “*neighborhood-based*” approach to defining notions of instance optimality. In a nutshell, we observe that most definitions for instance optimality boil down to choosing the “right” neighborhood. The smaller the neighborhood is, the stronger the notion of instance-optimality, and the harder it is to attain. If the neighborhood is the set of all instances, we are back to worst-case analysis. Thus the art is in choosing the smallest meaningful neighborhood that still allows for instance optimal algorithms. This general step-by-step process of increasing the neighborhood until one sees hope for instance-optimality reveals quite a bit about the fine-grained nature of a problem. Perhaps this way of looking at instance-optimality was already known, but since we have not seen it in print, we show (see Appendix C of the full version of this paper [15]) how most of the existing works on instance-

<sup>5</sup> [Afshani, Barbay and Chan [1]] state: “For example, consider the 2D convex hull problem, which has  $\Theta(n \log n)$  worst-case complexity in the algebraic computation tree model: for every input sequence of  $n$  points, one can easily design an algorithm  $A$  (with its code depending on the input sequence) that runs in  $O(n)$  time on that particular sequence, thus ruling out the existence of an instance-optimal algorithm.”

optimality, ranging from the classical works on adaptive sorting and presortedness [13], to later works by Demaine et al. [10] and Afshani et al. [1], and including the very recent work on universal optimality [18], all agree with this paradigm.

We define a neighborhood of the underlying DAG  $\vec{G}$ , denoted as  $\mathcal{N}(\vec{G})$ . See Definition 8 for a precise definition but intuitively, this is the set of all DAGs that are either isomorphic, or “one-flip” away from  $\vec{G}$ . Define the runtime  $T_A(\vec{G})$  of a randomized algorithm  $A$  on an instance  $\vec{G}$  as its expected comparison-cost on instance  $\vec{G}$ . Let  $\mathcal{C}(\vec{G})$  be the set of randomized (Las Vegas) algorithms that are correct for all instances in  $\mathcal{N}(\vec{G})$ . Now define

$$\text{OPT}(\vec{G}) = \inf_{A \in \mathcal{C}(\vec{G})} \max_{\vec{G}' \in \mathcal{N}(\vec{G})} T_A(\vec{G}').$$

For  $\alpha \geq 1$ , we say an algorithm  $A$  is  $\alpha$ -instance optimal if for every instance  $\vec{G}$ ,  $T_A(\vec{G}) \leq \alpha \text{OPT}(\vec{G})$ . Does there exist an algorithm achieving  $\alpha = O(1)$ , or  $\alpha = \log^{O(1)} n$ ?

### 2.1.2 InversionSort: An Almost Instance Optimal Algorithm

The algorithm we investigate for bipartite sorting is a variant of an algorithm InversionSort that was recently presented by us (Goswami and Jacob [16]) for the version when there is a sorted order on the nuts and bolts, and nuts can be compared to nuts (at a cost  $\alpha > 1$ ) and bolts can be compared to bolts (at a cost  $\beta > 1$ ). In bipartite sorting, two nuts (or two bolts) cannot be compared to each other, i.e.,  $\alpha = \beta = \infty$ , but due to the similarity to the previous algorithm we call the algorithm for bipartite sorting InversionSort too. Note that in [16] the previous algorithm was compared to the cost of the Hamiltonian, whereas now not only may the Hamiltonian cease to exist, its natural counterpart (the transitive reduction) is a weak lower bound.

As a first try, let us see what goes wrong when we apply the simple randomized QuickSort algorithm for nuts and bolts to bipartite sorting. We pick a random nut  $r$  and use it to pivot the bolts obtaining two sets  $B_{<r}$  and  $B_{>r}$ . Since no match was obtained, in the alternating step we can select a random bolt  $b$ , say from  $B_{>r}$ , and pivot the nuts, obtaining  $R_{<b}$  (containing  $r$ ) and  $R_{>b}$ . Instead of the two perfectly-partitioned subproblems obtained in nuts and bolts, we unfortunately now have three subproblems:  $(R_{<b}, B_{<r})$ ,  $(R_{<b}, B_{>r})$ , and  $(R_{>b}, B_{>r})$ .

If the nuts and bolts were perfectly interleaved, we show (see Appendix A.1 of the full version of this paper [15]) that a BFS-style Quicksort algorithm that we call BackboneSort sorts using  $O(n \log n)$  comparisons. BackboneSort works in a sequence of alternating phases, a nut-phase and a bolt-phase. In a nut phase it tries to refine the nut subproblems and makes progress over the three subproblems above in a round-robin fashion: for example, to make progress in  $(R_{<b}, B_{<r})$ , it would select a random bolt in  $B_{<r}$  and pivot  $R_{<b}$ , and vice-versa in a bolt-phase.

Unfortunately, in a general instance of bipartite sorting the nuts and bolts are not perfectly interleaved everywhere, and it turns out that BackboneSort can perform badly (see Theorem 29 of full version [15]). The reason for this behavior arises from selecting a “random” pivot in the subproblem: for the perfectly interleaved instance, its sub-instances are also perfectly interleaved (in particular all three subproblems  $(R_{<b}, B_{<r})$ ,  $(R_{<b}, B_{>r})$ , and  $(R_{>b}, B_{>r})$  can be shown to be roughly a constant fraction of the original problem with good probability), and then such a random pivot is guaranteed to be good, just like in randomized Quicksort. However, if the underlying instance is lop-sided (think of many nuts in  $R_{<b}$  sandwiched between two bolts in  $B_{<r}$ ), a random bolt is not a good pivot. To add to this complication, an instance may be lop-sided in one region and perfectly interleaved in another, and an instance-optimal algorithm should ideally be able to detect such a situation.



InversionSort proceeds similarly to BackboneSort, but instead of selecting random pivots, it performs a random comparison: when trying to make progress in  $(R_{<b}, B_{<r})$ , it will look for “inversions”, defined as a pair  $(r, b) \in R_{<b} \times B_{<r}$  such that  $b < r$ . When it finds such a pair, it uses them as pivots. The intuition is that regions having “easy” subinstances (like perfectly interleaved ones) will resolve fast, whereas those that are lop-sided, like all nuts less than all bolts, will take longer, but this is a necessary task in this sub-region.

The following theorem quantifies the performance of InversionSort. Here,  $N = n + m$  and Definition 8 is our precise definition of instance optimality, sketched in the previous section.

► **Theorem 1** (Instance Optimality of InversionSort). *There exists a constant  $c > 0$ , such that for every instance  $\mathcal{I}$ , the cost of InversionSort on  $\mathcal{I}$  is, with probability at least  $1 - 1/N$ , at most  $c(\log N)^3 \text{OPT}(\mathcal{I})$ , where  $\text{OPT}(\mathcal{I})$  is as in Definition 8.*

Thus InversionSort is  $O(\log^3 N)$ -instance optimal, with respect to a natural notion of instance-optimality that accounts for the one-flip neighborhood of the underlying DAG. We conjecture that InversionSort is actually  $O(1)$ -instance optimal with this notion<sup>6</sup>.

## 2.2 Unexpected Result: Generalized Sorting and Sorting with Priced Information

While Theorem 1 gives an algorithm for bipartite sorting that is instance-optimal, what can we say for DAG sorting? Unfortunately proving a polylog-instance optimality guarantee seems out of reach for now. The reason is that even the “sortable” case of DAG sorting, where the DAG has a directed Hamiltonian path, has a current best bound of  $\tilde{O}(n^{1.5})$  comparisons, or in other words, is a factor  $\tilde{O}(\sqrt{n})$  away from the Hamiltonian cost (which is  $n - 1$ ). This results from an interesting randomized algorithm by Huang, Kannan and Khanna [20] that uses the work by Kahn and Linial [22] on balancing extensions via the Brunn-Minkowski theorem.

For a DAG that is not sortable, we extend the algorithm of Huang, Kannan and Khanna to give an algorithm that performs  $\tilde{O}(\min(w n^{1.5}, n^2))$  comparisons and outputs the transitive reduction of  $\vec{G}$  (see Theorem 20). Here  $w$  denotes the width of the DAG, which is the size of its largest antichain (a set of incomparable elements). We now point out an unexpected consequence of this result.

The problem of *sorting with priced information* introduced by Charikar, Fagin, Guruswami, Kleinberg, Raghavan and Sahai, [8], is a generalization of the classical, unit-cost, comparison-based sorting, defined as follows. The input is a weighted undirected graph  $G$  on  $n$  vertices, with the cost  $c_{ij} \in \mathbb{R}_{\geq 0}$  on the edge  $e_{ij}$  indicating the cost to compare keys (represented by vertices)  $v_i$  and  $v_j$ . As before, edges not in  $G$  have cost  $\infty$  and cannot be queried, and a query on an edge  $e_{ij}$  reveals if  $v_i < v_j$  (indicated as  $\vec{e}_{ij}$ ) or  $v_i > v_j$  (indicated as  $\bar{e}_{ij}$ ).

Since the hidden Hamiltonian path  $\mathcal{H}$  is the cheapest proof, its cost which equals  $\sum_{i=1}^{n-1} c_{x(i)x(i+1)}$  is a lower bound. Here  $x(1) < \dots < x(n)$  is the sorted order. [8] propose finding algorithms that come as close to the cost of  $\mathcal{H}$  as possible. The *competitive ratio* is defined as the ratio of the cost of the algorithm to the cost of  $\mathcal{H}$ , and the goal is to find an algorithm with small competitive ratio.

Several special cases of cost functions have been studied, but for arbitrary costs, almost nothing is known about the above problem. Note that the  $\tilde{O}(n^{1.5})$  result by Huang, Kannan and Khanna [20] works when all costs are either 1 or  $\infty$ . What about the version with

<sup>6</sup> As evidence, we prove in Theorem 28 in Appendix A.2 of full version [15] that InversionSort solves the perfectly interleaved instance in an optimal  $O(n \log n)$  comparisons.

arbitrary costs? Many works state that the general-cost version is “arbitrarily bad” (Huang, Kannan and Khanna [20]), “bleak” or “hopeless” (Gupta and Kumar [17]). The only evidence for this is an  $\Omega(n)$  lower bound on the competitive ratio of any algorithm that *finds the maximum*. There is an example where the costs are either 0, 1,  $n$  or  $\infty$ , and one can show that any algorithm that finds the maximum element  $m$  must have cost  $\Omega(n)$  times that of the cheapest proof (of  $m$  being the maximum) (Charikar et al. [8], Hartline et al. [19], Gupta and Kumar [17]). While it certainly provided intuition, the  $\Omega(n)$  lower bound for maximum (with costs in  $\{0, 1, n, \infty\}$ ) was never extended to sorting.

This makes the  $\{0, 1, n, \infty\}$  version interesting due to three reasons:

- do instances in this cost regime contain an  $\Omega(n)$  lower bound for sorting too?
- it is the natural step-up from generalized sorting with costs in  $\{1, \infty\}$ , and
- this is the first instance with forbidden comparisons that requires an *instance-specific analysis* for the competitive ratio. For generalized sorting and for stochastic sorting<sup>7</sup>, the cost of the Hamiltonian is always  $n - 1$ , so one only has to bound the cost of an algorithm, without worrying about the underlying instance. However, when costs are in  $\{0, 1, n, \infty\}$  the cost of the Hamiltonian can range from 0 to  $n(n - 1)$ , and so an algorithm must adapt to the underlying instance.

Our second theorem addresses this cost version of sorting with priced information, showing that it cannot be the  $\{0, 1, n, \infty\}$  version that makes sorting hopeless!

► **Theorem 2.** *Consider the problem of sorting when every comparison has a cost in  $\{0, 1, F, \infty\}$ , for any  $F \geq n^{3/4}$ . There exists a polynomial time randomized algorithm whose competitive ratio is  $\tilde{O}(n^{3/4})$ , with high probability.*

The main ingredient in the proof of this theorem is the aforementioned  $\tilde{O}(\min(wn^{1.5}, n^2))$  comparisons algorithm for DAG sorting. Even though DAG sorting does not promise a Hamiltonian, it turns out to be useful because it can be used as a subroutine in a “greedy” algorithm for the  $\{0, 1, n, \infty\}$  cost version: obtain with cheapest cost the partial DAG formed by all costs 0 and 1 comparisons.

**Organization.** We state our problems precisely in Section 3. This is followed by our result on bipartite sorting (Theorem 1) in Section 4, and our result on sorting with priced information (Theorem 2) in Section 5.

### 3 Problem Definitions

We formally define the problem of bipartite sorting first, and then the problems of DAG sorting and sorting with priced information. We invite the reader to see Figure 2 for the landscape of these problems, their relations to each other, and how our results fit in this landscape.

► **Definition 3 (Bipartite Sorting).** *Input: A complete bipartite undirected graph  $G$  of unit costs on  $V = R \cup B$ . Only edges in  $G$  can be queried (at unit cost), and querying an undirected edge  $(u, v)$  has one of two outcomes,  $u < v$  (implying  $\vec{uv} \in \vec{G}$ ) or  $u > v$  (implying  $\vec{vu} \in \vec{G}$ ).  $N := |V|$ .*

<sup>7</sup> Also initiated by Huang, Kannan and Khanna [20], this is the version where the input graph  $G$  is random

The instance of bipartite sorting is defined by a partition of  $R$  (reds) and  $B$  (blues) into stripes  $(S_1, \dots, S_k)$  (Figure 1), i.e., by the relative order<sup>8</sup> between the reds and the blues. Note that  $S_i$ s are unordered sets. The DAG  $\vec{G}$  has, for all  $1 \leq i \leq k-1$ , an edge from every element in  $S_i$  to every element of the other color greater than it, i.e., to every element in  $\cup_{\ell \geq 0} S_{i+(2\ell+1)}$ .

*Output:* The sequence of stripes  $(S_1, \dots, S_k)$  (see figure 1). Equivalently, the transitive reduction of  $\vec{G}$ .

**DAG Sorting.** Let  $\mathcal{P}$  denote a partial order on a set of  $n$  elements, and let  $\vec{P}$  denote the transitively closed DAG on  $V = \{v_1, \dots, v_n\}$  indicating all order relations in  $\mathcal{P}$ . That is, the vertex  $v_i$  is identified with element  $i$ , the edge  $\vec{e}_{ij}$  between vertices  $i$  and  $j$  exists if  $v_i < v_j$ , the edge  $\overleftarrow{e}_{ij}$  exists if  $v_i > v_j$ , and no edge between  $v_i$  and  $v_j$  exists if elements  $i$  and  $j$  are incomparable. For convenience, let  $P$  denote the set of (undirected) edges in  $\vec{P}$  without their directions. Let  $w$  denote the width of  $\vec{P}$ .

► **Definition 4** (Implied and essential edge, transitive reduction [2]). *Given a DAG  $\vec{G} = (V, \vec{E})$ , an edge  $(u, v) \in \vec{E}$  is implied, if there is a directed path in  $\vec{G}$  from  $u$  to  $v$ . Otherwise,  $(u, v)$  is essential. The set of essential edges is called the transitive reduction of  $\vec{G}$ .*

Note that for every implied edge  $(u, v) \in \vec{E}$ , there is a directed  $u$  to  $v$  path of essential edges in  $\vec{G}$ . It turns out that the transitive reduction of a DAG is unique [2]. Let  $\vec{T}$  denote the transitive reduction of  $\vec{P}$ . Let  $\mathcal{T}$  denote the undirected version of  $\vec{T}$ .

► **Definition 5** (DAG Sorting, arbitrary costs). *Input: An undirected graph  $G = (V, E \subset P)$  with costs  $c_{ij} \in \mathbb{R}_{\geq 0}$  on edges. An oracle that answers, given an undirected edge  $e_{ij}$  of  $G$ , its orientation in  $\vec{P}$ .*

*Promise:*  $\mathcal{T} \subset E$ . That is, the queryable edges contain the edges of the transitive reduction.

*Output:*  $\vec{T}$ .

*Cost of an algorithm  $\mathcal{A}$ :* The total cost of the edges queried by  $\mathcal{A}$  on the instance  $\vec{P}$ . This will be denoted by  $\text{cost}(\mathcal{A}, \vec{G})$ , where  $\vec{G}$  is the directed version of  $G$  (also referred to as the instance from now on), and contains all the information about  $\vec{P}$ .

When  $c_{ij} = 1$  for all  $i, j$ , we call the problem simply DAG Sorting.

We will only care about the query cost of the algorithm, and while there may be a compact representation of  $\vec{T}$ , we ask the algorithm to output  $\vec{T}$  in its entirety for simplicity.

DAG sorting generalizes sorting with priced information, which we describe next. This problem was introduced by [8] in the broader context of querying with priced information, where one wants to compute a function  $f$  of  $n$  inputs, and querying an input has a certain cost associated to it. The competitive ratio is defined as the (worst case) ratio of the cost of the query strategy to the *cost of the cheapest proof of  $f$* . This work initiated a multitude of papers on priced information, studying problems like learning with attribute costs [23], stochastic boolean function evaluation [11], searching on trees [26, 25], and priced information in external memory [5].

► **Definition 6** (Sorting with Priced Information [8]). *Sorting with priced information is a special case of DAG sorting, when the partial order  $\mathcal{P}$  is a total order. In this case,  $\vec{P}$  is a tournament and therefore  $P$  is a complete graph.  $\vec{T}$  is simply a directed Hamiltonian path*

<sup>8</sup>  $S_1$  is the set of all sources in the DAG  $\vec{G}$ ; since red and blue elements can be compared with unit-cost,  $S_1$  must necessarily be of one color.  $S_i$  can be iteratively defined as the set of sources in  $\vec{G}$  after all stripes 1 to  $i-1$  have been deleted.

$\mathcal{H}$  in  $\vec{P}$ . The input  $G$  is any graph on  $n$  vertices containing the edges of  $\mathcal{H}$  without their directions, and the output is  $\mathcal{H}$ . The total cost of the edges queried by an algorithm  $\mathcal{A}$  on the instance  $\vec{G}$  will be denoted by  $\text{cost}(\mathcal{A}, \vec{G})$ .

**Competitive ratio for sorting with priced information.** The **competitive ratio** of  $\mathcal{A}$  (as defined in [8]) is  $\rho(\mathcal{A}, n) = \max_{\vec{G}} \text{cost}(\mathcal{A}, \vec{G}) / \text{cost}(\mathcal{H})$ , where the maximum is taken over all instances  $\vec{G}$  of  $n$  vertices with a total order. The goal is to find sorting algorithms with small competitive ratio. Here  $\text{cost}(\mathcal{H})$  is considered as a proxy for the complexity of the instance  $\vec{G}$ , as it is the cheapest proof. It certainly is a valid lower bound, for the edges on the Hamiltonian *must* be queried by any algorithm.

For example, when  $G$  is the complete unit-cost graph, MergeSort or QuickSort achieve a competitive ratio of  $\Theta(\log n)$  (the latter w.h.p. if randomized). Similarly, when  $G$  is unit-cost but not complete, the  $\tilde{O}(n^{1.5})$  cost algorithm algorithm by Huang, Kannan and Khanna [20] has a competitive ratio of  $\tilde{O}(\sqrt{n})$ . As mentioned, very little is known about the case when  $G$  has arbitrary costs.

We end with the remark that DAG sorting is closely related to a line of work initiated by Faigle and Turán,[14] called sorting a partial ordered set, or identifying a poset. This was followed up by several works such as [9] and [12]. For a survey on this line of work that also includes generalized sorting, we refer the reader to [7].

## 4 Results on Bipartite Sorting

This section is divided into three subsections. In Section 4.1 we formally state our definition of instance-optimality. In Section 4.2 we derive some lower bounds on OPT stemming from the definition of instance-optimality. Finally, in Section 4.3 we define InversionSort, and prove that it comes close to achieving instance-optimality (Theorem 1) by charging the comparisons performed by InversionSort to the derived lower bounds.

### 4.1 Defining Instance-Optimality

As mentioned in the introduction, the following instance of bipartite sorting shows that comparing the cost of an algorithm to the transitive reduction is hopeless.

► **Definition 7 (One-inversion Instance).** Let  $G = (R, B, E)$  be the undirected complete bipartite graph on  $|R| = |B| = n/2$ . Pick an arbitrary  $r \in R$ ,  $b \in B$ , let  $R_{-r} = R \setminus \{r\}$  and  $B_{-b} = B \setminus \{b\}$ . Define a DAG  $\vec{G}$  via its transitive reduction as follows.

$$TR(\vec{G}) = \{\vec{x}b : x \in R_{-r}\} \cup \{\vec{b}r\} \cup \{\vec{r}y : y \in B_{-b}\}.$$

The transitive reduction has size  $O(n)$ , but any algorithm must spend  $\Omega(n^2)$  comparisons to identify  $r$  and  $b$ . Thus the “cheapest proof” is too strong a benchmark. We now present our neighborhood-based approach for bipartite sorting. This approach is general, and in the Appendix C of the full version [15], we show how this neighborhood-based approach fits several works on instance-optimal algorithms for static problems, namely the works on adaptive sorting [13], on set intersection, union and difference [10], and the recent work on universal optimality [18].

We start with small neighborhoods and gradually increase them until there is no immediate obstruction to instance-optimality. Define  $\mathcal{N}_{\mathcal{A}}(\vec{G})$  as the set of DAGs (Automorphic) isomorphic to  $\vec{G}$  if all edges in  $\vec{G}$  are unit-cost, and cost-isomorphic<sup>9</sup> otherwise. Next, define

<sup>9</sup> Meaning there exists an isomorphism between the DAGs that preserves the costs.

the runtime of an algorithm on an instance  $\vec{G}$  as its maximum comparison-cost on any instance in  $\mathcal{N}_{\mathcal{A}}(\vec{G})$ . Define  $OPT_{\mathcal{A}}(\vec{G})$  as the smallest comparison-cost of any algorithm.

For unit-cost sorting,  $\vec{G}$  is a complete DAG, and  $|\mathcal{N}_{\mathcal{A}}(\vec{G})| = n!$ . It is easily seen that now  $OPT(\vec{G}) = \Omega(n \log n)$ , and the unnecessary  $\log n$  gap arising from comparing to the cheapest proof vanishes. This is also consistent with the fact that any  $O(n \log n)$  algorithm that ignores the sequence of keys and only treats them as a set is  $O(1)$  instance optimal in the order-oblivious setting.<sup>10</sup>

Moving on to the case when  $\vec{G}$  is not complete, we see that the above definition is not sufficient by the following observation. Consider the case when  $\vec{G}$  is a complete bipartite graph, with all edges going in the same direction, i.e., from one partition  $R$  to the other  $B$ . Now  $|\mathcal{N}_{\mathcal{A}}(\vec{G})| = 1$ , and any algorithm that knows that it is operating on  $\vec{G}$  has zero comparison cost<sup>11</sup>. However, any instance-unaware algorithm needs  $\Omega(|R||B|)$  comparisons to verify that the instance is indeed  $\vec{G}$ . This suggests that we need a larger neighborhood than just  $\mathcal{N}_{\mathcal{A}}(\vec{G})$ .

Let  $\vec{E}$  denote the set of edges in the transitive reduction of  $\vec{G}$ , also called essential edges (Definition 4). Define  $\mathcal{N}_E(\vec{G})$  as the set of DAGs that differ from  $\vec{G}$  in exactly one essential edge being flipped, and any other changes it may imply. Again, define the runtime of an algorithm on an instance  $\vec{G}$  as its maximum comparison-cost on any instance in  $\mathcal{N}_E(\vec{G})$ , and define  $OPT_{\mathcal{A}}(\vec{G})$  as the smallest comparison-cost of any algorithm. It is straightforward now to observe that if  $\vec{G}$  is sortable,  $OPT(S) \geq \text{cost}(\mathcal{H})$  and if  $\vec{G}$  is not sortable,  $OPT(S) \geq \text{cost}(TR(\vec{G}))$ . Thus, we recover both definitions of competitive ratio by considering this one-flip neighborhood.

The algorithms we consider here are randomized, and we hence want a definition of instance optimality that allows for randomization. The notion of “instance optimal in the random-order setting” of [1], based on Yao’s principle, compares implicitly to the optimal expected running time of a correct randomized algorithm (Las Vegas style). Our following definition does this directly:

► **Definition 8** ( $\alpha$ -Instance Optimality). Let  $\mathcal{N}(\vec{G}) = \mathcal{N}_E(\vec{G}) \cup \mathcal{N}_{\mathcal{A}}(\vec{G})$ . Define the runtime  $T_A(\vec{G})$  of a randomized algorithm  $A$  on an instance  $\vec{G}$  as its expected comparison-cost on instance  $\vec{G}$ . Let  $\mathcal{C}(\vec{G})$  be the set of randomized (Las Vegas) algorithms that are correct for all instances in  $\mathcal{N}(\vec{G})$ . Define  $OPT(\vec{G}) = \inf_{A \in \mathcal{C}(\vec{G})} \max_{\vec{G}' \in \mathcal{N}(\vec{G})} T_A(\vec{G}')$ . For some  $\alpha \geq 1$ , an algorithm  $A$  is called  $\alpha$ -instance optimal if for every instance  $\vec{G}$ ,  $T_A(\vec{G}) \leq \alpha OPT(\vec{G})$ .

## 4.2 Lower Bounds on OPT

We first refine our notion of instance-optimality to make it more amenable to deriving lower bounds.

► **Definition 9** (Instance Optimality Distribution). Let  $\mathcal{C}'(\vec{G})$  be the set of deterministic algorithms that are correct for all instances in  $\mathcal{N}(\vec{G})$ . Let  $\mathcal{D}(\vec{G})$  be the uniform distribution over  $\mathcal{N}_{\mathcal{A}}(\vec{G})$ . Define  $OPT(\vec{G}) = \inf_{A \in \mathcal{C}'(\vec{G})} \mathbb{E}_{\vec{G}' \sim \mathcal{D}(\vec{G})} T_A(\vec{G}')$ .

An application of Yao’s principle [28] shows that for any  $\vec{G}$ :  $OPT_D(\vec{G}) \leq OPT(\vec{G})$ . Note that the optimal algorithm is allowed to depend on  $\mathcal{I}$ . We remark that while the following lower bounds would be easy to prove for algorithms unaware of  $\mathcal{I}$  using adversary arguments, we prove this for  $OPT$  (Definition 9), which requires some extra care.

<sup>10</sup> All algorithms that exploit certain presortedness in the input necessarily exploit the input sequence of keys. This corresponds to a smaller neighborhood than the  $n!$  size automorphism neighborhood.

<sup>11</sup> Recall that we do not charge to write down the transitive reduction which has size  $O(|R||B|)$ , only the query cost.

## 23:12 Instance-Optimal Algorithm and Sorting

► **Lemma 10** (Transitive Reduction or Verification lower bound). *Let  $\mathcal{I}$  be an instance of bipartite sorting, let  $K \subset \vec{E}_{\mathcal{I}}$  be its transitive reduction, and define  $C_V = |K|$ . Then, any algorithm that is correct for all inputs from  $\mathcal{N}(\mathcal{I})$  must perform at least  $C_V$  comparisons.*

**Proof.** Assume there exists an algorithm  $\mathcal{A}$  that is correct for all instances in  $\mathcal{N}(\mathcal{I})$  simultaneously that performs at most  $C_V - 1$  comparisons on input  $I$ . This means that there must exist an edge  $e$  on the transitive reduction that is not verified by  $\mathcal{A}$ . As  $\mathcal{A}$  is deterministic, it would report  $\mathcal{I}$  as output even when the input had  $e$  flipped because all other edges have the same direction as in  $\mathcal{I}$ , as we will argue now: If this (non-flipped) edge is between two stripes of size one, the two endpoints merge into two other stripes, but no edges changes direction. If this edge is between two stripes and both of them have size at least 2, then we create one additional inversion without changing other directions. If one stripe has size one and the other has size at least two, one element is moved from the size-at-least-two stripe to a neighboring stripe. Again, no bichromatic edges (implied or not) change their direction. ◀

While the above lower bound is natural as it is the cheapest proof, Definition 9 now allows for the following lower bound that captures the complexity of instances where transitive reduction is too weak a measure (recall the instance in Definition 7).

► **Lemma 11** (Inversion finding lower bound). *Let  $\mathcal{I}$  be an instance of bipartite sorting with  $n \geq 2$  red and  $m \geq 2$  blue elements, where not all comparisons come out the same, and define*

$$C_I = \frac{nm}{\min(|\{(r, b) \in R \times B \mid r < b\}|, |\{(r, b) \in R \times B \mid r > b\}|)}.$$

*Under the distribution of Definition 9, any deterministic algorithm  $\mathcal{A}$  that does at most  $C_I/2$  comparisons must fail with probability (at least)  $1/8$ .*

**Proof.** Let  $\mathcal{D}$  be the uniform distribution over  $\mathcal{N}_A(\mathcal{I})$ . Remember that  $\mathcal{N}_A(\mathcal{I})$  contains all instances where the stripes are internally arbitrarily permuted. Observe  $|\mathcal{N}_A(\mathcal{I})| \geq 4$  by the bounds on  $n$  and  $m$ . W.l.o.g., assume  $|\{(r, b) \in R \times B \mid r < b\}| < |\{(r, b) \in R \times B \mid b < r\}|$ , i.e., the usual outcome of a comparison is  $b < r$  and the inversion is  $r < b$ . Let  $p = 1/C_I = |\{(r, b) \in R \times B \mid r < b\}|/nm \leq 1/2$  be the probability that a randomly chosen pair of elements is an inversion. By Yao's principle, let  $\mathcal{A}$  be a deterministic algorithm and think of it as a decision tree  $\mathcal{T}$  where nodes are red-blue comparisons and non-inversion go to the left, inversions go to the right. Each leaf of the tree is marked with an output (that declares which instance was represented by the input), or a failure output.

**Claim.** Let  $v_k$  be the node  $v$  that is reached by  $k$  comparisons returning “non-inversion” (i.e. the leftmost node of  $\mathcal{T}$  at depth  $k$ ). When input is drawn from  $\mathcal{D}$ , the node  $v_k$  is reached with probability at least  $1 - kp$ .

**Proof of Claim.** An input reaches  $v_k$  if  $k$  (potentially dependent) random experiments all came out as “non-inversion”, each having a probability  $1 - p$ . The claim follows from a union bound over the fail events. ◀

From the claim it follows that if  $v_k$  is a leaf for  $k \leq C_I/2$ , the algorithm must fail with probability at least  $1/4$ : Then  $kp \leq C_I/2 \cdot 1/C_I = 1/2$  and  $1 - kp \geq 1/2$ , so half of the inputs reach  $v_k$ . Because there are at least four inputs, at least two reach  $v_k$ , but it can only be labeled with one, the other(s), which stand for at least  $1/4$  of the inputs in  $\mathcal{N}_A(I)$ , make the algorithm fail. ◀

Finally, we will need to combine various lower bounds from different subproblems. Let  $\mathcal{I}$  be an instance and  $(S_1, \dots, S_k)$  its stripes (see Figure 1). Consider pairs of indices  $(a_1, b_1), \dots, (a_\ell, b_\ell)$ , where for all  $1 \leq j \leq \ell$ ,  $a_j$  and  $b_j$  both belong to  $\{1, \dots, k\}$ , and  $a_j < b_j < a_{j+1} < b_{j+1}$  for all  $j < \ell$ . For  $1 \leq j \leq \ell$  define the subinstance  $\mathcal{I}_j$  by the subgraph of  $\vec{G}_{\mathcal{I}}$  on the vertices  $V_j = \bigcup_{i=a_j}^{b_j} S_i$ .

► **Lemma 12** (Decomposition into Lower Bounds for Subproblems). *For  $1 \leq j \leq \ell$ , let  $\mathcal{I}_j$  be a subinstances of  $\mathcal{I}$  as above. Then  $\text{OPT}_D(\mathcal{I}) \geq \sum_{j=1}^{\ell} \text{OPT}_D(\mathcal{I}_j)$ .*

**Proof.** As we are working with Definition 9, we first have to check that an algorithm that is correct on  $\mathcal{N}(\mathcal{I})$  is actually correct on each  $\mathcal{N}(\mathcal{I}_j)$ . To this end observe that every edge flip in  $\mathcal{I}_j$  is also an edge flip in  $\mathcal{I}$ , and that any permutation of the labels/names in  $\mathcal{I}_j$  is also a permutation in  $\mathcal{I}$ .

Run the algorithm  $A$  on an instance  $\mathcal{I}'$  drawn from  $\mathcal{D}(\mathcal{I})$ , and let  $X$  be the random variable describing the number of comparisons of  $A$ . Define the random variables  $X_j$  to be the number of comparisons between vertices in subproblem  $\mathcal{I}'_j$ . Then  $\sum X_j \leq X$ .

Let's conceptually draw  $\mathcal{I}'$  from  $\mathcal{D}(\mathcal{I})$  by first finding a position in the input list (name) for all vertices not in  $\mathcal{I}_j$  and finally draw names for the vertices in  $\mathcal{I}_j$ . Now we can think of  $A$  as deterministic algorithm for  $\mathcal{I}_j$  by considering all comparisons not in  $\mathcal{I}_j$  as fixed, and we get  $E[X_j] \geq \text{OPT}_D(\mathcal{I}_j)$ . The statement of the lemma follows by linearity of expectation. ◀

### 4.3 InversionSort and its $O(\log^3 n)$ instance-optimal guarantee

A generic state of InversionSort will be defined using a *backbone*, which is a sequence of elements of alternating colors, called *representatives* or *pivots*. Each representative will be assigned a *bucket*, which is a set of elements of the same color that lie between the two neighboring representatives of the other color on the backbone.

InversionSort makes progress from one state to the next by performing three steps: a) finding an *inversion* (which is defined soon) between neighboring representatives on the backbone, b) inserting this inversion on the backbone, and c) pivoting with these two elements, thereby refining the buckets.

#### 4.3.1 Description of InversionSort

► **Definition 13** (Backbone, Representatives, and Buckets). *The backbone consists of a totally ordered, alternating list of representatives  $(u_0, u_1, u_2, \dots, u_{2k}, u_{2k+1}) = (r_0, b_1, r_2, \dots, r_{2k}, b_{2k+1})$ , where  $r_{2i} \in R$  and  $b_{2j+1} \in B$  with  $r_i < b_{i+1}$  and  $b_i < r_{i+1}$ . Here,  $r_0$  is an artificial red element that is smaller than all elements, and the last element  $b_{2k+1}$  is an artificial blue element that is larger than all elements. The representatives define the buckets  $(X_0, X_1, X_2, \dots, X_{2k}, X_{2k+1}) = (R_0, B_1, R_2, \dots, R_{2k}, B_{2k+1})$  by  $R_i = \{x \in R \mid b_{i-1} < x < b_{i+1}\} \setminus \{r_i\}$  and  $B_i = \{x \in B \mid r_{i-1} < x < r_{i+1}\} \setminus \{b_i\}$ . Here, as a convention, the representative is not included in the bucket. Again,  $R_0 = \{x \in R \mid x < b_1\}$  and  $B_{2k+1} = \{x \in B \mid r_{2k} < x\}$  are special cases.*

► **Definition 14** (active subproblems and buckets). *As long InversionSort did not create a certificate that there are no further inversions between  $x_i$  and  $x_{i+1}$ , the subproblem defined by the buckets  $X_i$  and  $X_{i+1}$  is called active, and so are  $X_i$  and  $X_{i+1}$ .*

Our previous work [16] now defines an *inversion*, which gives the algorithm its name. Consider adjacent representatives  $u_i$  and  $u_{i+1}$ , their corresponding adjacent buckets  $X_i$  and  $X_{i+1}$ , and a bichromatic pair  $(x, y)$  of elements  $x \in X_i$  and  $y \in X_{i+1}$ . Observe that  $x$  and

---

**Algorithm 1** Algorithm InversionSort.
 

---

**Require:** elements  $R$  red,  $B$  blue  
 create trivial backbone  $\mathcal{B}$  from  $R$  and  $B$ , see Definition 13  
 $\eta \leftarrow 0$   
**while** there is an active subproblem (see Definition 14) in  $\mathcal{B}$  **do**  
    $\eta \leftarrow \eta + 1$   
   **for** each active (see Definition 14) bucket  $s$  **do**  
     Sample one element  $x_s$   
   **for** each active subproblem between buckets  $s$  (left), and  $q$  (right) **do**  
     Test for inversion between  $x_s$  and  $x_q$   
   **for** each active subproblem  $X_i, X_{i+1}$  where  $\eta$  - mark (age)  $> |X_i||X_{i+1}|$  **do**  
     do all comparisons between  $X_i$  and  $X_{i+1}$   
     update the backbone and certificates accordingly  
     the subproblem is finished, i.e. no longer active  
   **for** each found inversion **do**  
     update the backbone, including splitting buckets and resampling pivots  
     mark new subproblems with round  $\eta$  as age

---

$y$  are not ordered by transitivity of the backbone. Because  $x$  and  $y$  are of different color, they can be compared. If  $y < x$ , the pair is called an *inversion*. This allows one to extend the backbone: we get  $u_i < y < x < u_{i+1}$ , which is a chain of actual comparisons between elements of alternating color.

*The only way to find an inversion in a bipartite setting (this is where the bichromatic setting is different) is to uniformly at random, from all pairs in  $X_i$  and  $X_{i+1}$ , pick  $x$  and  $y$ .* If the fraction of inversions is  $p$ , then the probability of finding an inversion is  $p$  and the expected number of trials to find one is  $1/p$ .

InversionSort starts by (trivially) having the backbone consist only of the artificial smallest red element  $r_0$  and largest blue element  $b_1$ , and  $R_0 = R$  and  $B_1 = B$ . For a given backbone  $(u_0, u_1, u_2, \dots, u_{2k}, u_{2k+1}) = (r_0, b_1, r_2, \dots, r_{2k}, b_{2k+1})$ , InversionSort first, for each pair  $X_i, X_{i+1}$  of adjacent buckets that have not yet found an inversion or a proof that there is no further inversion (i.e., reached the adjacent-stripe verification bound), in a round-robin manner, does one round of inversion-searching by randomly comparing pairs of elements in adjacent buckets. If this leads to an inversion, the inverted pair is saved and the algorithm moves to the next pair of adjacent buckets. At the end of the round, all identified inversions are considered and used to extend the backbone. Then InversionSort splits existing buckets by pivoting with new elements on the backbone. Because there is at most one pair of inversions between each two neighboring representatives on the backbone, each element is compared to at most two new representatives in each round.

This reestablishes the backbone and creates some new pairs of neighboring buckets, for which InversionSort initializes the inversion finding procedures. The algorithm stops once all neighboring pairs of buckets are shown to not have an inversion, i.e., the comparisons in the verification bound between neighboring stripes have been performed.

**Analysis.** [16] visualizes a run of InversionSort as a ternary (refinement) tree, where nodes correspond to subproblems. For an internal node  $v$ , there is a corresponding subinterval on the backbone defined by two consecutive pivots, say a blue pivot followed by a red pivot,  $b_v < r_v$ . If InversionSort finds an inversion  $y < x$  ( $x$  is blue and  $y$  is red) between  $b_v$  and  $r_v$ , then  $v$  has three children with the respective pivots  $(b_v, y)$ ,  $(y, x)$ ,  $(x, r_v)$ .



The random nature of the inversion searching of InversionSort, is made precise in Lemma 11 in [16]. However, a stronger version of this lemma applies for bipartite setting with the same proof.

► **Lemma 15** (Randomness in Inversion Finding, Stronger Version of [16, Lemma 11, p11]). *At any stage of the InversionSort, consider a successful inversion finding procedure, which finds an inversion  $y < x$  between representatives  $u_i < y < x < u_{i+1}$ . Say, w.l.o.g., that  $u_i$  is red and  $u_{i+1}$  is blue, and hence  $x$  is red and  $y$  is blue.*

1. *for any  $y \in X_{i+1}$ , conditioned on  $y$  being in the inversion,  $x$  is uniformly distributed among all the red elements in  $R_y = \{x \in X_i \mid y < x < u_{i+1}\}$ , and.*
2. *for any  $x \in X_i$ , conditioned on  $x$  being in the inversion,  $y$  is uniformly distributed in  $B_x = \{y \in X_{i+1} \mid u_i < y < x\}$ .*

This gives a bound on the height of the tree.

► **Lemma 16** (Height of the refinement tree [16, Theorem 5, p11]). *Let  $\mathcal{T}$  be the refinement tree of running InversionSort on an instance  $\mathcal{I}$  with  $N = n + m$  elements. With high probability in  $N$ , the height of  $\mathcal{T}$  is  $O(\log N)$ .*

**Handling Overlaps.** Because of the overlapping nature of the problem, InversionSort cannot easily focus on elements between neighboring representatives. For example, for the child indicated by pivots  $(y, x)$ , instead of only getting the reds and blues that actually lie in this range as input, InversionSort instead has to also work with the red elements contained in  $(b_v, y)$  and the blue elements inside  $(x, r_v)$ . This “spill-in” from the neighboring subintervals on the backbone needs to be analyzed.

As is argued in [16], the cost of bichromatic inversion search procedure of InversionSort is justified by the subinstance (part of the Hamiltonian) between the neighboring elements on the backbone. In Section 4.3.2, we will analyse this in the bipartite setting using the notion of instance optimality. However, if the spill-in for this subproblem is too large, inversion search is too costly. Hence, [16] identify subproblems that do not have too much spill-in from their neighbors, and call these subproblems *unaffected*. Inversion search in unaffected subproblems can be charged to their subinstance. More precisely, [16] show that at any time, with high probability, at least roughly a  $1/(\log N)^2$  fraction of all current problems are unaffected. Accounting over the whole tree, including the pivoting, introduces another  $\log N$  factor corresponding to the depth of the tree.

### 4.3.2 Putting everything together: Proof of Theorem 1

We now complete the proof of our main result for bipartite sorting, restated here for convenience.

► **Theorem 1** (Instance Optimality of InversionSort). *There exists a constant  $c > 0$ , such that for every instance  $\mathcal{I}$ , the cost of InversionSort on  $\mathcal{I}$  is, with probability at least  $1 - 1/N$ , at most  $c(\log N)^3 \text{OPT}(\mathcal{I})$ , where  $\text{OPT}(\mathcal{I})$  is as in Definition 8.*

We will show how to charge the comparisons performed by InversionSort to the lower bounds presented in Section 4.2. First, Lemma 15 and Lemma 16 imply that the refinement tree height  $h = O(\log n)$  and hence the pivoting cost is  $O(n \log n)$ . Second, the considerations about unaffected subproblems remains valid, there are at most  $O(\log^2 n)$  affected subproblems per unaffected subproblem. The inversion searching cost in each unaffected subproblem is justified by Lemma 11 if an inversion is found, otherwise by Lemma 10. Adding the extra  $O(\log N)$  factor (height of the tree), the next lemma completes the proof of Theorem 1.

► **Lemma 17.** *Let  $\mathcal{T}$  be the refinement tree of height  $h$  for a run of *InversionSort* on instance  $\mathcal{I}$ , and let  $V_{\mathcal{T}}$  be the set of nodes of  $\mathcal{T}$ . Then*

$$\sum_{v \in V_{\mathcal{T}}} \text{OPT}(\mathcal{I}_v) \leq 2h \text{OPT}(\mathcal{I})$$

**Proof.** It suffices to show that for each level  $\mathcal{L}$  of  $\mathcal{T}$  the inequality  $\sum_{v \in \mathcal{L}} \text{OPT}(\mathcal{I}_v) \leq 2 \text{OPT}(\mathcal{I})$  holds. Note that the subinstances  $\mathcal{I}_v$  of the same polarity in any  $\mathcal{L}$  do not share vertices. Hence the decomposition Lemma 12 is applicable and the lemma follows because there are two polarities. ◀

## 5 Result on Sorting with Priced Information: Lower bound does not extend

Recall that there exists an instance demonstrating the lower bound of  $\Omega(n)$  on the competitive ratio of any algorithm that finds the **maximum** of a set of  $n$  elements. Announced in the original paper by Charikar et al.[8], this was spelled out one year later by Gupta and Kumar [17]. The instance is simple to describe (see Appendix B of the full version [15]), and all comparisons in it have costs in  $\{0, 1, n, \infty\}$ . Although this example was never formally stated for sorting, its discovery seems to have dampened efforts to study (either better algorithms, or lower bounds for) the general version of sorting with priced information in the past 20 years.

In this section, we prove Theorem 2, that shows that the  $\Omega(n)$  lower bound for maximum with costs in  $\{0, 1, n, \infty\}$  cannot extend to sorting.

► **Theorem 2.** *Consider the problem of sorting when every comparison has a cost in  $\{0, 1, F, \infty\}$ , for any  $F \geq n^{3/4}$ . There exists a polynomial time randomized algorithm whose competitive ratio is  $\tilde{O}(n^{3/4})$ , with high probability.*

While counterintuitive at a first glance (after all, the cost to sort is at least the cost to find the maximum), the simple explanation is that the cheapest proof for sorting is also more expensive than that of the maximum. This opens up the problem of arbitrary-cost sorting once again - is there a  $\Omega(n)$  lower bound for sorting with arbitrary costs, or can our  $\tilde{O}(n^{3/4})$  algorithm be extended to an  $o(n)$  competitive algorithm for arbitrary costs?

Theorem 2 is achieved by *first developing an algorithm for DAG sorting*. Why consider the case of an unsortable DAG, when the DAG we have is sortable? Here is the reasoning. If we consider greedy algorithms for sorting with priced information, it is natural to try to discover as much of  $\vec{G}$  as possible with low-cost edges<sup>12</sup>. However, note that the sub-DAG  $\vec{G}_{\leq w}$  consisting of edges with cost at most  $w$  in  $\vec{G}$  may not be sortable, which is exactly the problem of DAG sorting.

We set up some notation first. Given an undirected complete graph  $G$  with costs in  $\{0, 1, F, \infty\}$ , let  $\vec{G}$  denote the underlying DAG that contains a directed Hamiltonian path. Define  $\vec{G}_0$  as the DAG obtained by revealing all cost 0 edges, observe that it may not have

<sup>12</sup>We remark that some time after we first uploaded a version of Theorem 2 to Arxiv, a preprint by [Jiang, Wang, Zhang and Zhang, Arxiv [21]] was uploaded, where the authors also use an algorithm for DAG sorting (they call it GPSC) parameterized by  $w$  and extend our  $\tilde{O}(n^{3/4})$  algorithm to obtain a  $\tilde{O}(n^{1-1/2W})$  competitive algorithm for sorting with priced comparisons with at most  $W$  distinct costs. For our setting when  $W = 4$  they re-derive our result separately as their main theorem would give a  $n^{7/8}$ -competitive ratio.

a Hamiltonian path, and let  $w_0$  be the width of  $\vec{G}_0$ . Similarly, denote by  $\vec{G}_{01}$  the DAG obtained by revealing all cost 0 and 1 edges;  $\vec{G}_{01}$  may not have a Hamiltonian path either, and let  $w_{01}$  be the width of  $\vec{G}_{01}$ . Finally, let  $k_1$  and  $k_F$  be the number of cost 1 and cost  $F$  edges on the Hamiltonian path in  $\vec{G}$ .

## 5.1 Algorithm details

The following is our proposed algorithm for the  $0, 1, F, \infty$  cost version of sorting with priced information. Below, we will abbreviate the algorithm by Huang, Kannan and Khanna [20] for the  $1, \infty$  setting, by HKK.

■ **Algorithm 2** Algorithm for  $0, 1, F, \infty$  cost.

---

**Require:** undirected graph  $G = (V, E, c)$  with costs  $c(e) \in \{0, 1, F, \infty\}$

**Ensure:** the total order (directed Hamiltonian)

Probe all cost zero edges

Run the following 4 algorithms in parallel, performing one comparison from each. If any of them discover the Hamiltonian, report the edges in the Hamiltonian path, and abort the other algorithms

- Set  $F = \infty$ . Run the HKK algorithm on the cost 1 edges, starting from  $\vec{G}_0$
  - Set  $F = 1$ . Run the HKK algorithm on the cost 1 edges, starting from  $\vec{G}_0$
  - Run Algorithm 3 on the cost 1 edges, starting from  $\vec{G}_0$
  - Find the 0-1 DAG using Theorem 20, use Algorithm 3 with cost  $F$  edges on it.
- 

The running time of the final algorithm will then be a minimum of four running times. We briefly explain the first two algorithms, and then explain in detail the last two. Recall that HKK algorithm runs when costs are 1 or  $\infty$ . On an input with costs in  $\{0, 1, F, \infty\}$ , the first algorithm pretends that cost  $F$  edges are forbidden too, i.e.,  $F = \infty$ , and probes whatever edges HKK would have probed from the cost 1 edges. Clearly, this will not find the Hamiltonian if it contains cost  $F$  edges, as they aren't queried. However, in the case that the Hamiltonian does not contain cost  $F$  edges, it will sort the input, and stops. The second algorithm does the opposite: it does not differentiate between cost  $F$  and cost 1 edges, and probes them if they would have been probed by the HKK algorithm. If run for long enough, this will find the Hamiltonian, and is stopped once it does so.

### 5.1.1 Algorithm 3 : Hamiltonian By Predecessors

The third algorithm in Algorithm 2 is Algorithm 3, which is also used as the second half of the fourth algorithm in Algorithm 2. This algorithm finds a Hamiltonian path in a partially revealed DAG. It utilizes Lemma 18, that generalizes binary search to searching for predecessors of a vertex in a DAG of width  $w$ . For two DAGs  $D'$  and  $D$  on the same set of vertices, we will write  $D' \subset D$  if all the directed edges in  $D'$  are also contained in  $D$ .

► **Lemma 18** (Hamiltonian by predecessor search). *Let  $D' \subset D$  be two DAGs on the vertex set  $V$  and assume that  $D$  contains a Hamiltonian path. Assume that the Hamiltonian path in  $D$  contains a set  $\vec{S}$  of  $k$  edges that are not in  $D'$ , and let  $S$  be the undirected version of  $\vec{S}$ . Let  $E$  be a set of edges that can be queried and assume  $S \subset E$ . Let  $w$  be the width of  $D'$ . Then,  $k + 1 \geq w$  and the Hamiltonian in  $D$  can be found with  $O(wk \log n)$  queries on edges in  $E$ .*

Below is the pseudocode for Algorithm 3. It uses in turn a predecessor searching subroutine that is captured by the following simple lemma. Lemma 18.

## 23:18 Instance-Optimal Algorithm and Sorting

► **Lemma 19** (Predecessor search in DAG). *Given a DAG  $D' = (V, E')$  of width  $w$ , and a vertex  $v \in V$ ,  $|V| = n$ , let  $D$  be the DAG obtained by extending  $D'$  by probing all edges involving  $v$ . Define  $P_v = \{u \mid (u, v) \text{ is in the transitive reduction of } D\}$ . There exists an algorithm that computes  $P_v$  with  $O(w \log n)$  queries, and runs in  $O(n^2)$  time.*

**Proof.** Observe that any chain in  $D'$  can contain at most one element of  $P_v$ , and hence  $|P_v| \leq w$ . Indeed, we can run one binary search on each of the  $w$  chains in  $D'$ , leading to at most  $w$  candidate predecessors. The number of queries is easily seen to be  $O(w \log n)$  after computing an optimal partitioning into chains in polynomial time. ◀

**Proof of Lemma 18.** By Dilworth's Theorem,  $D'$  can be partitioned into  $w$  chains. To show  $k + 1 \geq w$ , assume otherwise,  $w > k + 1$ , and let  $A$  be  $k + 2$  non-comparable vertices in  $D'$ . By the pigeonhole principle, there must be two vertices of  $A$  in the same of the  $k + 1$  stretches of cost 0 edges on the Hamiltonian, a contradiction to them being incomparable.

To prove that Algorithm 3 performs at most  $O(kw \log n)$  queries, observe that adding edges to  $D'$  does not increase its width. In the while loop, as long as  $D'$  is not the Hamiltonian path, let  $S$  be the first layer of a BFS traversal of the transitive reduction of  $D'$  with  $|S| \geq 2$ , and observe that  $S$  is an antichain and hence  $|S| \leq w$ . All vertices of  $S$  but one have their incoming edge on the directed Hamiltonian not yet revealed / queried: Assume there are two vertices  $b \neq d \in S$  and their predecessors  $a \prec b$  and  $c \prec d$  are both already in  $D$ . Then,  $b$  and  $d$  are not sources in  $D$ , and hence  $S$  must be the set of successors of a vertex  $v$ . Additionally, there is only a single source  $s$  in  $D$ , and the set  $\{x \mid x \prec v\}$  forms a chain in the transitive reduction of  $D$  starting in  $s$ . This means  $w \leq a < b$  and  $w \leq c < d$  contradicting them being different.

By the above arguments,  $D$  contains the Hamiltonian with only  $k - |S| + 1$  unrevealed edges missing. We used  $O(|S|w \log n)$  queries to reduce the number of unrevealed edges by  $|S| - 1$  for  $|S| \geq 2$ , hence each search creating a missing edge of the Hamiltonian, and this search must justify at most one additional such search. Hence, the total number of queries to arrive at the Hamiltonian is  $O(wk \log n)$ .

Binary searching for a vertex  $v$  into one of the  $w$  chains takes  $O(\log n)$  probes, and in  $O(w \log n)$  probes one is sure to have at least discovered one edge from the Hamiltonian, namely the incoming edge to  $v$ . This can then be repeated  $k$  times, revealing the Hamiltonian.

■ **Algorithm 3** Hamiltonian By Predecessors.

---

**Require:** undirected  $G = (V, E)$  defining which comparisons are allowed

**Require:** DAG  $D$  of already probed edges (initially the cost 0 edges)

**Ensure:** The updated  $D$  contains a directed Hamiltonian

create (and maintain) a chain decomposition  $C$  of the transitive reduction of  $D$

**while**  $D$  has width  $w > 1$ , i.e. is not the intended result **do**

**if** The transitive reduction of  $D$  has several sources **then**

    Let  $S$  be the set of these sources

**else**

    Let  $v$  be the lowest vertex with more than 1 successors

    Let  $S$  be the set of successors of  $v$

**for** each  $u$  in  $S$  **do**

    Find all predecessors of  $u$  in  $D$  (Lemma 19), adding answers to  $D$

    \\ there are at most  $w$  such predecessors

---

### 5.1.2 The fourth algorithm in Algorithm 2

We will develop another algorithm, that proceeds in two steps: a) compute only the 0-1 DAG,  $\vec{G}_{01}$ , and b) find the cost  $F$  edges ( $k_F$ -many of them) on the Hamiltonian path. Step b) is performed using Algorithm 3. If  $k_F = 0$ , step a) recovers the complete Hamiltonian path. Before we state the DAG sorting algorithm for step a), we note that it only needs to output the transitive reduction of  $\vec{G}_{01}$ .

► **Theorem 20.** *There is a poly-time randomized algorithm that w.h.p. solves DAG sorting for an instance  $\vec{G}$  with edge costs in  $\{0, 1, \infty\}$ , using  $O(\min(w n^{3/2} \log n, n^2))$  comparisons, where  $w$  is the width of  $\vec{G}$ .*

We defer the proof of Theorem 20 for now and analyze our algorithm assuming it.

## 5.2 Analysis of Algorithm 2

► **Lemma 21.** *Algorithm 2 incurs the following costs*

$$\begin{cases} O(\min(n^{1.5} \log n, w_0 k_1 \log n)) & \text{if } k_F = 0 \\ O(\min(F n^{1.5} \log n, w_{01} n^{1.5} \log n + F w_{01} k_F \log n)) & \text{if } k_F > 0 \end{cases}$$

**Proof.** If  $k_F = 0$ , the first algorithm that ignores cost  $F$  edges by setting  $F = \infty$  never probes a cost  $F$  edge, and finishes in  $O(n^{1.5} \log n)$  comparisons (this is the cost of the algorithm by Huang, Kannan and Khanna [20]). Since the DAG formed by cost 0 edges has width  $w_0$  and  $k_F = 0$ ,  $w_0 \leq k_1 + 1$ . The third step running Algorithm 3 finishes after at most  $O(w_0 k_1 \log n)$  comparisons, by Lemma 18.

If  $k_F > 0$ , the first term comes from running HKK after setting  $F = 1$ : the true cost of probing an edge is at most a factor  $F$  larger. Finally, step 4 of Algorithm 2 runs the algorithm in Theorem 20 first, incurring at most  $w_{01} n^{1.5} \log n$  many comparisons. With the 0-1 DAG obtained using Theorem 20, Algorithm 3 now inserts at most  $k_F$  many edges in the Hamiltonian, probing at most  $w_{01} \log n$  many edges for each. Every probe costs  $F$ , for a total of  $w_{01} n^{1.5} \log n + F w_{01} k_F \log n$ . ◀

### 5.2.1 Proof of $\tilde{O}(n^{3/4})$ competitive ratio of Algorithm 2

We claim that the competitive ratio is always bounded by  $O(n^{3/4} \log n)$ . Observe that the cost of the Hamiltonian is  $k_1 + F k_F$ . If  $k_1 = k_F = 0$ , the Hamiltonian has a cost of 0 and our algorithm finds it at cost 0. From now on, we assume not both of  $k_1$  and  $k_F$  are 0.

Consider the case  $k_F = 0$  first. Note that this implies that the width  $w_{01}$  of  $\vec{G}_{01}$  is 1. First consider the subcase when  $w_0 \leq n^{3/4}$ . In this case, the competitive ratio is bounded above by  $O(w_0 k_1 \log n) / k_1 = O(w_0 \log n) \leq O(n^{3/4} \log n)$ . In the subcase when  $w_0 > n^{3/4}$ , observe that this implies that  $k_1 \geq n^{3/4}$  which implies that the competitive ratio is bounded above by  $O(n^{1.5} \log n) / k_1 \leq O(n^{3/4} \log n)$ .

Next, consider the case  $k_F \geq 1$ , and the cost of the Hamiltonian is at least  $F k_F$ . Since  $w_{01} \leq k_F + 1$ , the cost of the algorithm is at most  $O(w_{01} n^{1.5} \log n + F w_{01} k_F \log n) < O(k_F n^{1.5} \log n + F w_{01} k_F \log n)$ , and dividing by  $F k_F$  (the lower bound on the cost of the Hamiltonian), we get a competitive ratio of at most  $O((n^{1.5}/F + w_{01}) \log n)$ . Since  $F \geq n^{3/4}$ , this ratio is  $O(n^{3/4} \log n)$  as long as  $w_{01} \leq n^{3/4}$ . Else if  $w_{01} > n^{3/4}$ , we observe that  $k_F \geq n^{3/4}$ , and then the  $F n^{1.5} \log n$  query cost gives us a competitive ratio of at most  $F n^{1.5} \log n / F k_F \leq n^{3/4} \log n$ . Thus Theorem 2 is proved. ◀

It remains to prove Theorem 20, which is the topic of the next subsection.

### 5.3 Proof of Theorem 20

First, observe that if the width of  $\vec{G}$  is at least  $\sqrt{n}/4$ , the statement of Theorem 20 is easy to achieve by probing all edges. Hence, let us assume the width is at most  $\sqrt{n}/4$ . We will show that there is an algorithm that computes  $\vec{G}_{01}$  with cost at most  $O(w_{01}n^{1.5} \log n)$ . This algorithm will only probe cost 0 and 1 edges, and will be a generalization of the algorithm in [20]. Note that while the algorithm in [20] works on a cost  $\{1, \infty\}$  setting under the promise of a Hamiltonian path in the true graph, our algorithm finds the transitive reduction of the DAG  $\vec{G}_{01}$  of width  $w_{01}$ . We first address the challenges posed in extending the work by Huang Kannan and Khanna [20].

**Challenges in extending the results of Huang, Kannan and Khanna [20].** At a high level, the algorithm in [20] alternates between three ways of making progress:

1. Finding and probing balanced edges, defined as those that reduce the number of possible linear extensions of the current DAG by a  $1 - (1/(e\sqrt{n}))$  factor. Finding such edges requires approximating the average rank of vertices under all possible linear extensions at all stages of the algorithm.
2. After estimating the indegree of vertices upto an additive error of  $\tilde{O}(\sqrt{n})$  by an  $\tilde{O}(n^{1.5})$  sampling procedure, the algorithm probes free edges, defined as the set of edges  $(u, v)$  where the average rank of  $u$  is smaller than the average rank of  $v$ , and  $v$  has most  $\tilde{O}(\sqrt{n})$  unprobed incoming edges. Free edges that are balanced again reduce the number of linear extensions by a constant factor. Otherwise, they can contribute at most  $\tilde{O}(n^{1.5})$  to the total cost.
3. Binary Search - When there are no free edges, there must exist a set of  $\sqrt{n}$  vertices with known total order (Lemma 3.5 in [20]). The other vertices can perform binary search into these  $\sqrt{n}$  vertices at a cost of  $O(n \log n)$ , and doing so removes these  $\sqrt{n}$  vertices from the picture. The total cost of binary search is therefore  $\tilde{O}(n^{1.5})$ .

*The third step of the algorithm is the step that guarantees a reduction in the problem size. However, the third step of this algorithm no longer works for DAG sorting: the existence of a set of  $\sqrt{n}$  vertices with known total order crucially relies upon the existence of the Hamiltonian path.*

**Proof of Theorem 20.** All of the definitions, algorithms, and accounting to estimate the in-degree of a vertex in [20] remain valid and unchanged. Observe that any topological sorting of the underlying directed graph, together with the undirected graph, reveal the directed graph. Define the **average rank of a vertex** as the average rank over all linear extensions of the true underlying directed graph. The following result implies that the average rank  $r$  on a path in (the transitive reduction of) a DAG is increasing by at least one per edge.

► **Lemma 22.** *Let  $D = (V, E)$  be a DAG and  $r: V \rightarrow \mathbb{Q}^{\geq 0}$  be the average rank. Then for  $(u, v) \in E$  we have  $r(u) + 1 \leq r(v)$ .*

**Proof.** Let  $\Pi$  be the set of all linear extensions that are compatible with  $D$ , such that  $r(x)|\Pi| = \sum_{\pi \in \Pi} \pi(x)$ . Then  $|\Pi|(r(v) - r(u)) = \sum_{\pi \in \Pi} \pi(v) - \pi(u) \geq |\Pi| \cdot 1$ . ◀

► **Definition 23** (Convex vertex subset). *In a DAG  $G = (V, E)$ , a subset of vertices  $S \subseteq V$  is convex if for every pair of vertices  $u, v \in S$ , every vertex  $w$  on any directed path from  $u$  to  $v$  in  $G$  is also in  $S$ .<sup>13</sup>*

Hence, considering a subset of the vertices by an upper and a lower bound on the average rank, leads to a convex subset. Next, a vertex is **live** if there is an unprobed edge incident to it, otherwise it is **exhausted**. The **assumed graph** is the same directed graph as in [20]. An **active vertex** is one that has at most  $4\sqrt{n} \log n$  unprobed in-edges in the assumed graph. A **free edge** is an unprobed edge  $(u, v)$  where the endpoint  $v$  is active. The proof of the next lemma is identical to that in [20].

► **Lemma 24** (Generalization of Lemma 3.5 in [20]). *The  $\sqrt{n}$  live vertices with smallest average rank are all active.*

► **Lemma 25** (Generalization of Lemma 3.6 in [20]). *If there are no free edges, and the width of the underlying  $\vec{G}$  is at most  $\sqrt{n}/4$ , then there exists a set  $S$  of at least  $(3/4)\sqrt{n}$  live vertices with known partial order who form a DAG of width at most  $\sqrt{n}/4$ .*

**Proof.** Consider the set  $S$  of at most  $\sqrt{n}$  live vertices with smallest average rank. More precisely, we chose the largest upper bound on the average rank such that  $|S| \leq \sqrt{n}$ . By Lemma 22, at most  $\sqrt{n}/4$  vertices can have the same average rank, such that  $|S| \geq 3/4\sqrt{n}$ . By Lemma 24, all vertices of  $S$  are active. Let  $u, v \in S$  be a pair of vertices that have a directed path  $P$  from  $u$  to  $v$  in  $\vec{G}$ . Then, all of this path  $P$  is in  $S$ , and all live vertices of  $P$  are in  $S$ . Hence, because there are no free edges, and all vertices of  $P$  not in  $S$  are exhausted, all edges of  $P$  must be probed. Hence,  $S$  is convex. The statement on the width follows from a chain decomposition of  $\vec{G}$  remaining a chain decomposition for a convex subset of vertices. ◀

Note that Lemma 25 does not imply that the algorithm can, or should, identify precisely this set  $S$  defined in the proof. Hence, the algorithm is going to approximate the smallest width subset of at least  $3/4\sqrt{n}$  vertices among the live vertices. More precisely, starting from an empty  $S$ , it is going to iteratively find the longest (outside  $S$ ) chain among the live vertices (also using edges that are implied by transitivity via edges in  $S$ ). It stops once  $S$  contains at least  $3/4\sqrt{n}$  vertices, and uses it as the DAG of small width in the setting of Lemma 19, and determine for every remaining live vertex its predecessors in  $S$ , with  $O(w \log n)$  queries each, compared to the  $O(\log n)$  queries if there is a Hamiltonian. Hence, the total number of queries increases from  $O(n^{3/2} \log n)$  to  $O(w n^{3/2} \log n)$ , as claimed in Theorem 20, completing the proof.

---

## References

- 1 Peyman Afshani, Jérémy Barbay, and Timothy M Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1):1–38, 2017.
- 2 Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

---

<sup>13</sup>Our definition of convexity differs from the definition in the metric graph theory (defined on undirected graphs), where convex subgraph contains the vertices of only the shortest paths between every pair of vertices [4].

- 3 Noga Alon, Manuel Blum, Amos Fiat, Sampath Kannan, Moni Naor, and Rafail Ostrovsky. Matching nuts and bolts. In *Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, pages 690–696, 1994.
- 4 Hans-Jürgen Bandelt and Victor Chepoi. Metric graph theory and geometry: a survey. In *Surveys on Discrete and Computational Geometry: Twenty Years Later*, editors, *Jacob E. Goodman and János Pach and Richard Pollack*, volume 453 of *Contemporary Mathematics*, pages 49–86. AMS, 2008.
- 5 Michael A Bender, Mayank Goswami, Dzejlja Medjedovic, Pablo Montes, and Kostas Tsichlas. Batched predecessor and sorting with size-priced information in external memory. In *Latin American Symposium on Theoretical Informatics*, pages 155–167. Springer, 2021.
- 6 Jean Cardinal, Justin Dallant, and John Iacono. An instance-optimal algorithm for bichromatic rectangular visibility. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 7 Jean Cardinal and Samuel Fiorini. On generalized comparison-based sorting problems. In *Space-Efficient Data Structures, Streams, and Algorithms*, pages 164–175. Springer, 2013.
- 8 Moses Charikar, Ronald Fagin, Venkatesan Guruswami, Jon Kleinberg, Prabhakar Raghavan, and Amit Sahai. Query strategies for priced information. *Journal of Computer and System Sciences*, 64(4):785–819, 2002.
- 9 Constantinos Daskalakis, Richard M Karp, Elchanan Mossel, Samantha J Riesenfeld, and Elad Verbin. Sorting and selection in posets. *SIAM Journal on Computing*, 40(3):597–622, 2011.
- 10 Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 743–752, 2000.
- 11 Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1453–1466. SIAM, 2014.
- 12 Devdatt P Dubhashi, Kurt Mehlhorn, Desh Ranjan, and Christian Thiel. Searching, sorting and randomised algorithms for central elements and ideal counting in posets. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 436–443. Springer, 1993.
- 13 Vladimir Estivill-Castro and Derick Wood. A survey of adaptive sorting algorithms. *ACM Comput. Surv.*, 24(4):441–476, 1992. doi:10.1145/146370.146381.
- 14 Ulrich Faigle and Gy Turán. Sorting and recognition problems for ordered sets. *SIAM Journal on Computing*, 17(1):100–113, 1988.
- 15 Mayank Goswami and Riko Jacob. On instance-optimal algorithms for a generalization of nuts and bolts and generalized sorting, 2023. arXiv:2211.04601.
- 16 Mayank Goswami and Riko Jacob. An algorithm for bichromatic sorting with polylog competitive ratio. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.
- 17 Anupam Gupta and Amit Kumar. Sorting and selection with structured costs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 416–425. IEEE, 2001.
- 18 Bernhard Haeupler, Richard Hladík, Václav Rozhoň, Robert Tarjan, and Jakub Tětek. Universal optimality of dijkstra via beyond-worst-case heaps, 2023. arXiv:2311.11793.
- 19 J Hartline, E Hong, A Mohr, E Rocke, and K Yasuhara. Personal communication. In , 2000.
- 20 Zhiyi Huang, Sampath Kannan, and Sanjeev Khanna. Algorithms for the generalized sorting problem. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS'11)*, pages 738–747, 2011. doi:10.1109/FOCS.2011.54.
- 21 Shaofeng H. C. Jiang, Wenqian Wang, Yubo Zhang, and Yuhao Zhang. Algorithms for the generalized poset sorting problem, 2023. arXiv:2304.01623.



- 22 Jeff Kahn and Nathan Linial. Balancing extensions via brunn-minkowski. *Combinatorica*, 11(4):363–368, 1991.
- 23 Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC'05)*, pages 356–365, 2005.
- 24 János Komlós, Yuan Ma, and Endre Szemerédi. Matching nuts and bolts in  $O(n \log n)$  time. *SIAM Journal on Discrete Mathematics*, 11(3):347–372, 1998.
- 25 Shay Mozes, Krzysztof Onak, and Oren Weimann. Finding an optimal tree searching strategy in linear time. In *In Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, volume 8, pages 1096–1105, 2008.
- 26 Krzysztof Onak and Pawel Parys. Generalization of binary search: Searching in trees and forest-like partial orders. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 379–388. IEEE, 2006.
- 27 Gregory JE Rawlins. *Compared to what? An introduction to the analysis of algorithms*. Computer Science Press, Inc., 1992.
- 28 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.



# Learning-Augmented Maximum Independent Set

Vladimir Braverman 

Rice University, Houston, TX, USA  
Google Research

Prathamesh Dharangutte 

Rutgers University, NJ, USA

Vihan Shah 

University of Waterloo, ON, Canada

Chen Wang 

Rice University, Houston, TX, USA  
Texas A&M University, College Station, TX, USA

---

## Abstract

---

We study the Maximum Independent Set (MIS) problem on general graphs within the framework of learning-augmented algorithms. The MIS problem is known to be NP-hard and is also NP-hard to approximate to within a factor of  $n^{1-\delta}$  for any  $\delta > 0$ . We show that we can break this barrier in the presence of an oracle obtained through predictions from a machine learning model that answers vertex membership queries for a fixed MIS with probability  $1/2 + \varepsilon$ . In the first setting we consider, the oracle can be queried once per vertex to know if a vertex belongs to a fixed MIS, and the oracle returns the correct answer with probability  $1/2 + \varepsilon$ . Under this setting, we show an algorithm that obtains an  $\tilde{O}(\sqrt{\Delta}/\varepsilon)^1$ -approximation in  $O(m)$  time where  $\Delta$  is the maximum degree of the graph. In the second setting, we allow multiple queries to the oracle for a vertex, each of which is correct with probability  $1/2 + \varepsilon$ . For this setting, we show an  $O(1)$ -approximation algorithm using  $O(n/\varepsilon^2)$  total queries and  $\tilde{O}(m)$  runtime. <sup>2</sup>

**2012 ACM Subject Classification** Theory of computation → Graph algorithms analysis; Computing methodologies → Machine learning

**Keywords and phrases** Learning-augmented algorithms, maximum independent set, graph algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.24

**Category** APPROX

**Related Version** Full Version: <https://arxiv.org/abs/2407.11364>

**Funding** Vladimir Braverman: Supported partially by the Naval Research (ONR) grant N00014-23-1-2737, and NSF-CNS 2333887 award.

Prathamesh Dharangutte: Supported by NSF through IIS-2229876 and CCF-2118953.

Vihan Shah: Supported in part by Sepehr Assadi's Sloan Research Fellowship and NSERC Discovery Grant.

**Acknowledgements** The authors are grateful to Sepehr Assadi and Samson Zhou for the helpful conversations regarding the project. The authors also thank anonymous APPROX reviewers for helpful suggestions.

## 1 Introduction

We consider learning-augmented *maximum independent set* (MIS) in this paper. Given a (unweighted, undirected) graph  $G = (V, E)$ , an independent set is a set of vertices  $I \subseteq V$ , such that for any  $u, v \in I$ ,  $(u, v) \notin E$ , i.e., there is *no* edge between  $u$  and  $v$ . The maximum independent set problem asks to find the independent set with the largest size in  $G$ .

---

<sup>1</sup> Throughout we use  $\tilde{O}(\cdot)$  to hide polylog( $n$ ) factors.

<sup>2</sup> A full version appears on arxiv under the same title.



Finding the maximum independent set is one of the classical NP-hard problems [42]. Furthermore, the seminal work of [36, 60] demonstrates the NP-hardness of approximating the size of the MIS to within a factor of  $n^{1-\delta}$  for any  $\delta > 0$ . In contrast, outputting any single vertex gives an  $n$ -approximation trivially. [10] gave a non-trivial  $O(n/\log^2 n)$ -approximation to MIS and this was later improved by [29]. These results indicate that the problem is quite hard in its general form and thus, many research efforts have been devoted to approximation algorithms in special settings, e.g., planar graphs [3, 47], rectangle-intersection graphs [16, 22, 32], and exponential-time algorithms [51, 31, 59, 12].

On the other hand, heuristic algorithms, despite their bad worst-case guarantees, often exhibit commendable performance on real-world graphs [4, 24, 57]. For instance, the greedy algorithm only offers an approximation guarantee of  $O(\Delta)$ , where  $\Delta$  is the *maximum degree* of  $G$ . However, it frequently yields satisfactory empirical results. The gap between the worst-case hardness and practical efficiency motivates us to study the MIS problem through the lens of beyond worst-case analysis [11, 52]. In particular, under the modern context, we ask the question of finding the maximum independent set with *learning-augmented oracles*.

### Learning-augmented algorithms

Learning-augmented algorithms, also known as algorithms with predictions, have attracted considerable attention in recent years (see, e.g. [50, 38, 46, 21, 56, 7, 9, 1, 37, 13, 53], and references therein). This paradigm of beyond worst-case analysis has been successful in surmounting classical thresholds and bridging the gap between the worst-case hardness and practical efficiency (see, e.g., [48], for an excellent summary). Typically, in learning-augmented algorithms, we assume the access to an oracle that gives part of the “right answer” to the problem, and fails with some small but non-negligible probability. Conceptually, these algorithms aim to take advantage of modern machine learning models, which are fairly accurate on predictive tasks yet make random mistakes in an inconsistent fashion. Learning-augmented algorithms provide a great way to analyze algorithms beyond the worst case, and these algorithms usually have immediate implications in practice (see the empirical results in, e.g., [20, 38, 26, 56, 1]). Inspired by the recent work in utilizing machine learning-based techniques for the maximum independent set [2, 49, 14], we consider the MIS problem through the lens of learning-augmented algorithms.

The advantage of the learning-augmented algorithms has inspired a flurry of work that studies *graph problems* within this framework [8, 30, 17, 18, 39, 5, 45, 6, 23, 33]. In a very recent work, [23] considered the Max-cut problem, in which the oracle model is closely related to our setting for the MIS problem. Under the Unique Game Conjecture (UCG), it is known that getting anything better than  $\alpha \approx 0.878$  approximation for max-cut is NP-hard ([43]). In contrast, [23] showed that with a learning-augmented oracle, we could achieve better approximation than the  $\alpha$  threshold in polynomial time. In another closely related work, [33] studied the more general constraint satisfaction problems (CSPs) through the lens of the learning-augmented algorithms. There, they obtain results for both the Max-cut and the Max 2-Lin problem. Although [23, 33] defines more general learning-augmented oracles, they, unfortunately, fall short of capturing the MIS-type of CSP problems, and their results do not have direct implications on the MIS problem.

From the above discussion, we can see that *a)* studying the maximum independent set problem in the framework of learning-augmented algorithms has great potential; and *b)* to this end, the existing models and algorithms are not yet sufficient. In light of this, we ask the following question:

*Under the framework of learning-augmented algorithms, what efficient algorithms can we get for the maximum independent set problem?*

## 1.1 Our models and contributions

In what follows, we will define the learning-augmented oracle model we consider and present our main results.

### Our oracle model

We consider the following natural learning-augmented oracle: for a fixed maximum independent set  $I^*$ , the oracle answers whether a vertex  $v \in I^*$  correctly with probability  $1/2 + \varepsilon$ , and incorrectly with probability  $1/2 - \varepsilon$ . In addition, the randomness is *independent* across the vertices. We denote by  $\text{ORC}_{G,I^*}(v)$  the answer the oracle gives when queried for vertex  $v$ .

We study approximation algorithms for MIS with the learning-augmented oracle in two settings: the *persistent noise* setting and the *non-persistent noise* setting. We discuss the settings and the results, respectively, as follows.

- The **persistent noise** setting. In the persistent noise setting, the randomness of  $\text{ORC}_{G,I^*}$  is drawn exactly *once*. Therefore, the answer for a vertex will remain the same no matter how many times you query the oracle. Another way to think about this is that the oracle can be queried at most once for a vertex. This setting is the most standard in the learning-augmented literature, and graph problems are often studied under persistent noise (see, e.g. [30, 17, 18, 58, 39, 5, 23, 33] and references therein). Our main result in this setting is a randomized algorithm that with high probability<sup>3</sup> achieves an  $\tilde{O}(\sqrt{\Delta})$  (multiplicative) approximation to the MIS in  $O(m)$  time (Theorem 1).
- The **non-persistent noise** settings. In this setting, for each vertex  $v$ , we allow  $\text{ORC}_{G,I^*}(v)$  to use *fresh randomness* for different queries. If we are allowed to make  $O(n \log n)$  queries to the oracle in total, then we can trivially recover the entire set  $I^*$  with high probability by querying each vertex  $O(\log n)$  times. The interesting case is when we are allowed to make only  $O(n)$  queries, i.e., a number that is *asymptotically the same* as the persistent noise setting. Although the non-persistent noise setting is less frequently studied in the learning-augmented algorithm literature, it has recently sparked considerable interest in various problems [34, 35, 44]. In Appendix B of the full version, we show that it is easy to get an  $O(\log n)$ -approximation with  $O(n)$  queries. Our main result considerably improved on the approximation factor: we show that we can indeed obtain an  $O(1)$  approximation with  $O(n)$  queries and  $\tilde{O}(m)$  runtime (Theorem 5).

Our results in the persistent noise setting hold assuming full independence, but it can be easily extended to the setting where oracle queries are assumed to use  $k$ -wise independent hash function for  $k = O(\log n)$ . Extending it to the pair-wise independent case is challenging as the failure probabilities in the concentration bounds are not enough for the application of a union bound.

## 1.2 Technical overview

The biggest challenge in leveraging the oracle information is distinguishing the case where  $\text{ORC}_{G,I^*}(v)$  is indeed correct. In what follows, we give a high-level overview of our techniques describing how we can use the neighborhood information for this purpose. For the simplicity of the discussion, we always assume  $\varepsilon = \Theta(1)$  in the technical overview.

<sup>3</sup> As standard in the literature, we use “with high probability” to denote a success probability of  $1 - 1/\text{poly}(n)$ .

### Persistent noise setting

A natural approach in this setting would be to figure out the conditions in which a “yes” signal for a vertex  $v$  from the oracle implies  $v \in I^*$ , by aggregating signals from  $N(v)$ . However, such an idea is hard in the following sense. For a vertex  $v$  whose oracle query  $\text{ORC}_{G,I^*}(v) = 1$ , if there are many  $u \in N(v)$  such that  $\text{ORC}_{G,I^*}(u) = 1$ , we can determine that  $v$  should *not* be in the MIS. However, the converse is not true: if a vertex  $v$  is *not* in  $I^*$ , it does *not* necessarily have many neighbors in  $I^*$ . As a result, simply aggregating neighborhood information might not be enough to determine the membership of a vertex in the MIS.

The key idea here is, instead of looking at the oracle answer for vertex  $v$  ( $\text{ORC}_{G,I^*}(v)$ ), we look at what the oracle says for the *neighborhood* of the vertex  $v$ . This turns out to be a good enough signal to eliminate vertices that have *many* edges to the MIS  $I^*$ . Specifically, we can show that if  $v$  has  $\tilde{\Omega}(\sqrt{\deg(v)})$  edges to  $I^*$ , then the oracle queries for  $N(v)$  contain enough information to identify such a vertex  $v$ . Upon removal of such vertices, the remaining vertices have a small degree ( $\tilde{O}(\sqrt{\Delta})$ ) to  $I^*$ , and a greedy independent set on the residual vertices gives a good approximation.

### Non-persistent noise setting

Our algorithm for this setting is a bit more nuanced as we aim to minimize the query complexity to the oracle while aiming to achieve a good approximation. The starting point of our algorithm is from the viewpoint of the classical *pure exploration* algorithms in *multi-armed bandits (MABs)*. If we ignore the nature of MIS in our oracle, we can reduce to the following MABs problem: given  $n$  arms with mean rewards as either  $\frac{2}{3}$  or  $\frac{1}{3}$ , find *all* the arms with mean reward  $\frac{2}{3}$  with  $O(n)$  arm pulls. It is well-known that one can find a *single* best arm with high constant probability in  $O(n)$  queries. The question is, can we solve the problem by resorting to purely MABs algorithms, and simply ignoring the nature of the MIS?

It turns out that the above plan is not generally feasible. In particular, we note that returning the set of *all* arms with the higher reward is very similar to finding the *top- $k$  arms* in the MABs literature (see, e.g. [40, 41, 15, 19, 55]). In general, it would require  $\Omega(n \log k)$  arm pulls to obtain top- $k$  arms with high constant probability ([41, 55]). In Appendix C of the full version, we provide lower bound results, showing that to find even  $O(1)$  fraction of the high-reward arms in the instance distribution requires  $\omega(n)$  queries. The lower bounds teach us that to obtain the desired query efficiency and approximation guarantee, we have to exploit the structure of the MIS.

To better understand the hardness and the insights of MABs algorithms on our problem, let us look at the elimination-based algorithm as in the classical algorithm of [27, 28]. The first idea we can try is to adapt the elimination algorithm to our problem. To this end, a natural idea is to perform elimination based on whether the mean empirical reward of an arm is more than  $\frac{1}{2}$ . More concretely, we maintain a pool  $\tilde{I}$  of surviving vertices and use  $s_r$  as the number of queries to each vertex in round  $r$  with  $s_1 = O(1)$ . In round  $r$ , we can query  $\text{ORC}_{G,I^*}(v)$  for  $s_r$  time for each  $v \in \tilde{I}$ . We then eliminate all vertices  $v \in \tilde{I}$  whose number of “yes” answers is less than  $s_r/2$ , and recurse on the new  $\tilde{I}$  to round  $r + 1$ , for which we set  $s_{r+1} = 1.5s_r$ .

Since the probability for any  $v \notin I^*$  to survive decreases doubly-exponentially with the number of rounds, we can show that *all* vertices  $v \notin I^*$  are eliminated after  $O(\log \log n)$  rounds, and the total sample complexity on the *non-MIS* vertices is at most  $O(n)$ . Furthermore, the probability of losing any  $v \in I^*$  decreases exponentially, we can argue that in the end,  $\tilde{I}$

contains at least  $\Omega(1)$  fraction of the vertices in  $I^*$ . Unfortunately, due to this fact, for each vertex  $v \in I^* \cap \tilde{I}$ , i.e., the vertices in the MIS that survive till the end, we need to pay for  $2^{O(\log \log n)} = \text{polylog } n$  on the sample complexity. Therefore, this pure exploration algorithm only works when the size of  $I^*$  is upper-bounded by  $n/\text{polylog } n$ , and its worst-case guarantee is only a  $\text{polylog } n$  approximation.

Note that a  $\text{polylog } n$  approximation is far from what we want: after all, there is a trivial algorithm that achieves  $O(\log n)$  approximation with  $O(n)$  samples (see Appendix B of full version for details). Nevertheless, the existence of such an algorithm teaches us that the problematic case is when the MIS size is large and, in particular, *comparable* to the size of the non-MIS vertices. As such, a natural idea is to design an algorithm that handles the case when the numbers of the MIS and the non-MIS vertices are comparable, and fuse this algorithm with the elimination-based MABs procedure we discussed above.

The above idea is quite close to the final strategy we adapt, albeit we proceed differently for the roles of the two components. In particular, we use the pure exploration MABs algorithm not to output a set with vertex set  $\tilde{I} \subseteq I^*$ , but to output a set of vertex set  $\tilde{I}$  whose *majority (but not necessarily all)* of vertices are in  $I^*$ . To this end, we use a more *conservative* elimination strategy than the ones in the line of [27, 28]: instead of increasing the number of samples by a multiplicative factor, we increase the number of samples in each round by an *additive* factor. In this way, we cannot guarantee that all the “wrong” arms are eliminated; however, we can argue that, since the probability for the non-MIS vertices to survive decreases exponentially, we have *i)* the number of samples used on the vertices in  $I^*$  is bounded by  $O(n)$  *before* the size of  $\tilde{I} \setminus I^*$  reduces to the size of  $\tilde{I} \cap I^*$ ; and *ii)* the number of vertices in  $\tilde{I} \cap I^*$  only decreases by a constant fraction. In this way, we can design an efficient procedure that eliminates the “surplus” non-MIS vertices to always create cases when the number of non-MIS vertices is smaller.

The final missing piece is the MIS algorithm that deals with the case when the number of MIS vertices takes the majority of the vertex set. Our algorithm to handle this case is to compute an *approximate vertex cover* of the graph and the remaining vertices will form an approximate independent set. It is a well-known fact that if we compute a *maximal matching* and take *all* their endpoints, it forms a 2-approximate vertex cover that covers all edges in the graph. Furthermore, since the size of the non-MIS vertices is small, there can be only a *limited number* of vertices  $v \in I^*$  that can be counted in the vertex cover. As such, we can simply *remove* these vertices from the graph. The rest of the graph would form an independent set, and since we remove at most a constant fraction of vertices from  $I^*$  throughout the two phases, we get an  $O(1)$  approximation.

## 2 Preliminaries

### Notation

For a graph  $G = (V, E)$ , we use  $\deg(v)$  and  $N(v)$  for each vertex  $v \in V$  to denote the degree and neighborhood of  $v$ , respectively. We use  $G[U]$  for any set  $U$  of vertices to denote the induced subgraph of  $G$  on  $U$ .

We let  $I^*$  denote a fixed maximum independent set of the graph  $G$ . We let  $N_{I^*}(v) = N(v) \cap I^*$  be the set of neighbors of the vertex  $v$  in the independent set and let  $\deg_{I^*}(v) := |N_{I^*}(v)|$  be its size. Furthermore, we let  $\tilde{N}_{I^*}(v)$  be the set of neighbors of the vertex  $v$  that are claimed to be in the independent set by the oracle and let  $\tilde{\deg}_{I^*}(v)$  be its size.

For the purpose of conciseness, we defer the technical preliminaries to Appendix A.

### 3 An Algorithm in the Persistent Noise Setting

In this section we present an algorithm for the learning-augmented MIS problem with persistent noise. Formally we prove the following

► **Theorem 1.** *There exists a randomized algorithm that given*

- i) *an input graph  $G = (V, E)$  with maximum degree  $\Delta$  and*
- ii) *an MIS oracle  $\text{ORC}_{G, I^*}$  with persistent noise for an unknown maximum independent set  $I^*$ ,*

*in  $O(m)$  time outputs an independent set  $I$  such that  $|I| \geq \frac{\varepsilon}{12} \cdot (\Delta \ln n)^{-0.5} \cdot |I^*|$  with high probability.*

We dedicate the remainder of this section to the proof of Theorem 1. We start with the assumption that  $\varepsilon \leq 1/4$  (we can do this for any constant  $> 0$ ). This assumption is needed for technical reasons. If  $\varepsilon > 1/4$ , then it is easy to simulate an oracle with  $\varepsilon = 1/4$  by flipping the oracle answer with probability  $p = \frac{\varepsilon - 1/4}{1/2 + \varepsilon}$  ( $p \geq 0$  since  $\varepsilon > 1/4$ ). If we do this then the probability that the oracle gives the incorrect answer is  $(1/2 - \varepsilon) + p \cdot (1/2 + \varepsilon) = 1/4$  which is exactly what we wanted. Note that the final bound we get on the approximation factor now changes by a factor of at most 2. This is because when  $\varepsilon > 1/4$  we are replacing it with an oracle for  $\varepsilon = 1/4$  and the approximation factor linearly depends on  $\varepsilon$ .

#### The algorithm and analysis

We now state our algorithm.

■ **Algorithm 1** An algorithm for MIS in persistent noise setting.

---

**Input:** A graph  $G = (V, E)$  with maximum degree  $\Delta$  that contains an unknown maximum independent set  $I^*$ ; an MIS oracle  $\text{ORC}_{G, I^*}$  in the persistent noise setting

**Output:** A set of vertices  $I$  such that  $I$  forms an independent set and  $|I| \geq \frac{\varepsilon}{3} \cdot (\Delta \ln n)^{-0.5} \cdot |I^*|$ .

**Parameters:**  $s_v := (1/2 - \varepsilon) \deg(v) + 6\sqrt{\ln n} \cdot (1/2 - \varepsilon) \sqrt{\deg(v)}$ .

1. Calculate  $\widetilde{\deg}_{I^*}(v)$  for all vertices  $v \in V$ .
  2. Let  $L$  be the set of vertices where  $\widetilde{\deg}(v) \leq 36 \ln n$  for  $v \in V$ .
  3. Let  $S$  be the set of vertices where  $\widetilde{\deg}_{I^*}(v) \leq s_v$  for  $v \in V \setminus L$ .
  4. Output the greedy MIS  $I$  on  $G[S \cup L]$ .
- 

We first show that if  $v \in I^*$ , the number of “yes” answers in  $N(v)$  cannot be too high.

▷ **Claim 2.** If  $v \in I^* \setminus L$  then with high probability,  $\widetilde{\deg}_{I^*}(v) \leq (1/2 - \varepsilon) \deg(v) + 6\sqrt{\ln n} \cdot (1/2 - \varepsilon) \sqrt{\deg(v)}$ .

*Proof.* If  $v \in I^*$  then  $\deg_{I^*}(v) = 0$  which means that the expected size of  $\widetilde{\deg}_{I^*}(v)$  is  $(1/2 - \varepsilon) \deg(v)$ . Since we assume complete independence for the oracle we can use the Chernoff bound to get concentration.

Let  $X_i = 1$  if  $i^{\text{th}}$  neighbor is claimed to be in  $I^*$  by the oracle where  $i \in [\deg(v)]$ . Observe that  $\widetilde{\deg}_{I^*}(v) = \sum_i X_i$  is the number of neighbors that claim to be in  $I^*$ . We know  $\mu = \mathbb{E}[\widetilde{\deg}_{I^*}(v)] = (1/2 - \varepsilon) \deg(v)$ . Using the Chernoff (Proposition 12) bound with  $\delta_v = 6 \left( \frac{\ln n}{\deg(v)} \right)^{0.5} \leq 1$ :

$$\Pr \left( \widetilde{\deg}_{I^*}(v) > (1 + \delta_v) \mu \right) \leq \exp \left( -\frac{\delta_v^2 \cdot \mu}{3} \right) \leq n^{-3}. \quad (\text{since } \varepsilon \leq 1/4)$$

Notice that as  $\deg(v)$  gets larger we get better concentration. ◁



Note that Claim 2 does *not* rule out the case that a vertex  $v \in V \setminus I^*$  and has very few neighbors in  $I^*$ . Nevertheless, it tells us that if we simply eliminate the vertices that “block” a large number of neighbors in  $I^*$ , we will not mistakenly drop vertices in  $I^*$ .

Next, we show that if a vertex  $v$  has many neighbors in  $I^*$  i.e.  $\deg_{I^*}(v)$  is large then  $\widetilde{\deg}_{I^*}(v)$  should also be large and hence we should be able to detect such a vertex  $v \notin I^*$ .

▷ **Claim 3.** If  $v \notin I^*$  and  $\deg_{I^*}(v) \geq (3/\varepsilon)\sqrt{\ln n}\sqrt{\deg(v)}$  then with high probability,  $\widetilde{\deg}_{I^*}(v) > (1/2 - \varepsilon)\deg(v) + 6\sqrt{\ln n} \cdot (1/2 - \varepsilon)\sqrt{\deg(v)}$ .

Proof. If  $v \notin I^*$  and  $\deg_{I^*}(v) = k$  then the expected size of  $\widetilde{\deg}_{I^*}(v)$  is

$$\mu = \mathbb{E} \left[ \widetilde{\deg}_{I^*}(v) \right] = k(1/2 + \varepsilon) + (\deg(v) - k)(1/2 - \varepsilon) = (1/2 - \varepsilon)\deg(v) + 2\varepsilon k.$$

We now use the Chernoff bound (Proposition 11) with  $t = \varepsilon k$  to get concentration:

$$\begin{aligned} \Pr \left( \widetilde{\deg}_{I^*}(v) < \mu - t \right) &\leq \exp(-2t^2 / \deg(v)) \\ &= \exp(-2\varepsilon^2 k^2 / \deg(v)) \\ &\leq n^{-3}. \end{aligned} \quad (\text{using the lower bound on } k)$$

Thus, with high probability we have:

$$\begin{aligned} \widetilde{\deg}_{I^*}(v) &\geq \mu - \varepsilon k \\ &= (1/2 - \varepsilon)\deg(v) + \varepsilon k \\ &= (1/2 - \varepsilon)\deg(v) + 3\sqrt{\ln n}\sqrt{\deg(v)} \\ &> (1/2 - \varepsilon)\deg(v) + 6\sqrt{\ln n} \cdot (1/2 - \varepsilon)\sqrt{\deg(v)}. \end{aligned} \quad \triangleleft$$

We can conclude that the events in Claim 2 and Claim 3 happen with high probability by a union bound over all vertices.

**Finalizing the proof of Theorem 1.** Calculating  $\widetilde{\deg}_{I^*}(v)$  for all vertices  $v \in V$  and finding set  $S$  takes  $O(m)$  time. The greedy MIS can also be computed in  $O(m)$  time.

We first condition on the events in Claim 2 and Claim 3 for all vertices (this happens with high probability). Notice that for all vertices in  $v \in S$  we have  $\widetilde{\deg}_{I^*}(v) \leq s_v$ . By Claim 2 all vertices in  $I^*$  are in  $S$ . By Claim 3 we know that any non-MIS vertices  $v$  that are in  $S$  have  $\deg_{I^*}(v) \leq (3/\varepsilon)\sqrt{\ln n}\sqrt{\deg(v)} \leq (6/\varepsilon)\sqrt{\Delta \ln n}$ . Also, vertices in  $L$  have  $\deg_{I^*}(v) \leq \deg(v) = \sqrt{\deg(v)} \cdot \sqrt{\deg(v)} \leq \sqrt{\Delta}\sqrt{36 \ln n} \leq (6/\varepsilon)\sqrt{\Delta \ln n}$ .

This means that when we run the greedy MIS algorithm and pick a non-MIS vertex, we eliminate at most  $(6/\varepsilon)\sqrt{\Delta \ln n}$  vertices in  $I^*$ . Thus, we have  $|I| \geq \frac{\varepsilon}{6} \cdot (\Delta \ln n)^{-0.5} \cdot |I^*|$ . Finally, because of the assumption on  $\varepsilon$  ( $\varepsilon \leq 1/4$ ), we lose a factor of at most 2 in the approximation, giving us the final bound  $|I| \geq \frac{\varepsilon}{12} \cdot (\Delta \ln n)^{-0.5} \cdot |I^*|$ . ◀

▶ **Remark 4.** We assume that we have complete independence between the oracle queries for the vertices. But we can get essentially the same result (up to constants) when the oracle answers the queries using a  $k$ -wise independent hash function instead of a completely random function for  $k = O(\log n)$ .

This holds because we use Proposition 13 with  $k = O(\log n)$  instead of the Chernoff bound (Proposition 12). The min in the exponent always picks the second term because  $k$  is large enough and so we get something very similar to the Chernoff bound in Proposition 12 where the exponent only differs by some constants. Thus, the approximation we get will be a small constant factor worse but will remain the same asymptotically.

#### 4 An Algorithm in the Non-persistent Noise Setting

In this section, we consider algorithms in the *non-persistent noise setting* (*MABs setting*) of the MIS oracle, i.e., the algorithm can access the learning-augmented MIS oracle with *fresh randomness* for each query of a vertex  $v$ . The formal statement of our main result in this setting is as follows.

► **Theorem 5.** *There exists a randomized algorithm that given a parameter  $\delta \in (0, 1)$  and*

- i) *an input graph  $G = (V, E)$  with a maximum independent set  $I^*$ ; and*
- ii) *an MIS oracle  $\text{ORC}_{G, I^*}$  in the non-persistent noise setting,*

*with probability at least  $(1 - \delta)$ , in  $O(m \log n)$  time and  $\frac{30n}{\varepsilon^2} \cdot \log \frac{1}{\delta}$  total queries to  $\text{ORC}_{G, I^*}$ , computes a set  $I$  such that  $|I| \geq \frac{48}{50} \cdot |I^*|$ .*

We dedicate the remainder of this section to the proof of Theorem 5.

#### The algorithm

As we have discussed in our high-level overview, our algorithm proceeds in two phases. In the first phase, our algorithm focuses on eliminating most of the vertices in the non-MIS vertex set. Then, in the second phase, we show that a good approximation to vertex cover is enough to get a good approximation to the independent set. We can easily find a 2-approximate vertex cover in  $O(m)$  time by computing a maximal matching and picking all its endpoints. The detailed description of the algorithm is as follows.

■ **Algorithm 2** An algorithm for MIS in non-persistent noise setting.

---

**Input:** A graph  $G = (V, E)$  that contains an unknown maximum independent set  $I^*$ ; an MIS oracle  $\text{ORC}_{G, I^*}$  in the multi-armed bandit setting; a confidence parameter  $\delta \in (0, 1)$ .

**Output:** A set of vertices  $I$  such that  $I$  forms an independent set and  $|I| = O(|I^*|)$ .

**Parameters:**  $q_r = \frac{4}{\varepsilon^2} \cdot (r + \log \frac{1}{\delta})$ .

- Maintain a set of  $V_r$  with the initialization  $V_0 \leftarrow V$ .
  - For  $r = 1$  to  $\infty$ , do the following:
    1. **Elimination phase:**
      - For each vertex  $v \in V_{r-1}$ :
        - a. Query  $v$  for  $q_r$  times.
        - b. Remove  $v$  from  $V_{r-1}$  if the number of 1 returned by  $\text{ORC}_{G, I^*}(v)$  (“yes” answers) is less than  $q_r/2$ .
      - Let the updated vertex set be  $V_r$ , i.e.,  $V_r$  is a subset of vertices of  $V_{r-1}$  that gets at least  $q_r/2$  “yes” answers from  $\text{ORC}_{G, I^*}(v)$ .
    2. **Vertex Cover phase:**
      - a. Compute a 2-approximate vertex cover  $U_r$  of the induced subgraph  $G[V_r]$ .
      - b. Let  $I_r \leftarrow V_r \setminus U_r$ .
    3. Maintain the set  $I$  with the maximum size among all  $I_r$ 's, i.e., let  $I \leftarrow I_r$  if  $I_r$  is larger than  $I$  and keep  $I$  unchanged otherwise.
    4. If the total number of  $\text{ORC}_{G, I^*}$  queries is more than  $30 \cdot \frac{n}{\varepsilon^2} \cdot \log \frac{1}{\delta}$  then terminate and return the currently maintained  $I$ .
- 

Note that since we do *not* necessarily know the actual size of  $I^*$ , we compute a vertex cover after every elimination phase and simply output the independent set with the largest size throughout the process.

### The analysis

We now proceed to the analysis of the algorithm. Before diving into the main lemmas, we first show some straightforward technical claims that characterize the behavior of the MIS and non-MIS vertices in the elimination phase. We first show that the probabilities of an MIS vertex being eliminated and a non-MIS vertex surviving in round  $r$  are both small.

▷ **Claim 6.** The following statements are true:

1. Let  $v \in V_{r-1} \cap I^*$ ; then, the probability that  $v$  is removed from  $V_r$  is at most  $\frac{1}{100} \cdot \frac{\delta}{4^r}$ .
2. Let  $v \in V_{r-1} \setminus I^*$ ; then, the probability that  $v$  is *not* removed from  $V_r$  is at most  $\frac{1}{100} \cdot \frac{\delta}{4^r}$ .

*Proof.* We prove this claim by applying the Chernoff bound in Proposition 11. For any vertex  $v \in I^*$ , let the random variable  $X_v^i = 1$  if the  $i^{\text{th}}$  query for vertex  $v$  is a “yes” and  $X_v^i = 0$  otherwise for  $i \in [q_r]$ . Observe that  $X_v = \sum_i X_v^i$  is the number of “yes” answers returned by  $\text{ORC}_{G, I^*}(v)$  out of the  $q_r$  queries. Clearly, we have that  $\mathbb{E}[X_v] = (1/2 + \varepsilon) \cdot q_r$ , and  $X_v$  is a summation of the independent indicator random variables so, we can apply Proposition 11 to show that

$$\begin{aligned}
 \Pr\left(X_v < \frac{q_r}{2}\right) &= \Pr(X_v - \mathbb{E}[X_v] \leq -\varepsilon \cdot q_r) \\
 &\leq \exp(-2 \cdot \varepsilon^2 \cdot q_r) && \text{(applying Proposition 11)} \\
 &= \exp\left(-8r - 8 \log \frac{1}{\delta}\right) && \text{(by the definition of } q_r) \\
 &\leq \exp(-6) \cdot \exp(-2r) \cdot \exp\left(-8 \log \frac{1}{\delta}\right) \\
 &\leq \frac{1}{100} \cdot \frac{\delta}{4^r}.
 \end{aligned}$$

Note that the vertices in  $v \in I^* \cap V_{r-1}$  are in  $I^*$ . Therefore, we can get the desired statement for  $v \in I^* \cap V_{r-1}$ .

We can similarly define  $Y_v$  for the number of “yes” answers returned by  $\text{ORC}_{G, I^*}(v)$  with  $q_r$  queries for a vertex  $v \in V \setminus I^*$ . Here, we have that  $\mathbb{E}[Y_v] = (1/2 - \varepsilon)q_r$ . As such, we have that

$$\begin{aligned}
 \Pr\left(Y_v \geq \frac{q_r}{2}\right) &= \Pr(Y_v - \mathbb{E}[Y_v] \geq \varepsilon \cdot q_r) \\
 &\leq \exp(-2 \cdot \varepsilon^2 \cdot q_r) && \text{(applying Proposition 11)} \\
 &= \exp\left(-8r - 8 \log \frac{1}{\delta}\right) && \text{(by the definition of } q_r) \\
 &\leq \exp(-6) \cdot \exp(-2r) \cdot \exp\left(-8 \log \frac{1}{\delta}\right) \\
 &\leq \frac{1}{100} \cdot \frac{\delta}{4^r}.
 \end{aligned}$$

This gives us the desired statement for  $v \in V_{r-1} \setminus I^*$  as well. ◁

We now prove the main technical lemma of our algorithm that helps eventually prove Theorem 5. In what follows, we will denote the size of  $I^*$  as  $\alpha n$  for some  $\alpha \in (0, 1)$ . Our main lemma for the elimination phase is as follows.

► **Lemma 7.** *Let  $|I^*| = \alpha n$  for some  $\alpha \in (0, 1)$  and  $\tilde{r} = 1 + \log \frac{1}{\alpha}$ . With probability at least  $1 - \delta$ , the following statements about Algorithm 2 are true:*

- 1) *The number of vertices in  $V_{\tilde{r}}$  that are not in  $I^*$  is at most  $\alpha n/100$ , i.e.,*

$$|V_{\tilde{r}} \setminus I^*| \leq \frac{\alpha n}{100}.$$

## 24:10 Learning-Augmented Maximum Independent Set

II) The number of vertices in  $V_{\tilde{r}}$  that are in  $I^*$  is at least  $49/50 \cdot \alpha n$ , i.e.,

$$|V_{\tilde{r}} \cap I^*| \geq \frac{49}{50} \cdot \alpha n.$$

III) The total number of  $ORC_{G,I^*}$  queries in the first  $\tilde{r}$  rounds is at most  $30n/\varepsilon^2 \cdot \log 1/\delta$ , i.e.,

$$\sum_{r=1}^{\tilde{r}} |V_{r-1}| \cdot q_r \leq 30 \cdot \frac{n}{\varepsilon^2} \cdot \log \frac{1}{\delta}.$$

Note that in the above,  $|V_{r-1}| \cdot q_r$  is exactly the number of queries used in round  $r$ .

**Proof.** We prove the statements in order.

**Proof of i).** Note that by Claim 6, the probability that a vertex in  $V \setminus I^*$  survives round  $r$  is at most  $\frac{1}{100} \cdot \frac{\delta}{4^r}$ . As such, we have that

$$\begin{aligned} \mathbb{E}[|V_{\tilde{r}} \setminus I^*|] &= \sum_{v \in V_{\tilde{r}-1} \setminus I^*} \Pr(v \text{ survives round } \tilde{r}) \\ &= \sum_{v \in V \setminus I^*} \Pr(v \text{ survives all rounds till } \tilde{r}) \\ &= \sum_{v \in V \setminus I^*} \prod_{i=1}^{\tilde{r}} \Pr(v \text{ survives round } i \mid v \text{ survives all rounds till } i-1) \\ &\hspace{15em} \text{(All rounds are independent)} \\ &\leq \sum_{v \in V \setminus I^*} \prod_{i=1}^{\tilde{r}} \frac{\delta}{100} \cdot \frac{1}{4^i} \\ &\leq n \cdot \left(\frac{\delta}{100}\right)^{\tilde{r}} \cdot \left(\frac{1}{4}\right)^{\binom{\tilde{r}}{2}} \\ &\leq \frac{\delta n}{100} \cdot \left(\frac{1}{4}\right)^{\tilde{r}} \\ &\leq \frac{\alpha \cdot n \cdot \delta}{400}. \hspace{10em} \text{(using } \alpha \in (0, 1)) \end{aligned}$$

Therefore, by Markov inequality, we have

$$\Pr\left(|V_{\tilde{r}} \setminus I^*| > \frac{\alpha n}{100}\right) \leq \frac{\delta}{4}$$

as desired.

**Proof of ii).** By Claim 6, the probability that a vertex  $v$  is eliminated in round  $r$  is at most  $\frac{\delta}{100} \cdot \frac{1}{4^r}$ . We analyze the number of vertices in  $I^*$  that are eliminated by round  $r$ . We can show that the expected value is

$$\begin{aligned} \mathbb{E}[|I^* \setminus V_{\tilde{r}}|] &= \sum_{v \in I^*} \Pr(v \text{ is eliminated by round } \tilde{r}) \\ &\leq \sum_{v \in I^*} \sum_{i=1}^{\tilde{r}} \Pr(v \text{ is eliminated in round } i) \hspace{5em} \text{(Union Bound)} \\ &\leq \sum_{v \in I^*} \sum_{i=1}^{\tilde{r}} \frac{\delta}{100} \cdot \frac{1}{4^i} \\ &\leq (\alpha n) \cdot \frac{\delta}{100} \cdot \frac{1}{3}. \hspace{10em} \text{(Geometric Sum)} \end{aligned}$$

Therefore, by a simple Markov bound, we have that

$$\Pr\left(|I^* \setminus V_r| > \frac{\alpha n}{50}\right) \leq \frac{\delta}{6}.$$

Thus, with probability at least  $1 - \delta/6$  we have  $|I^* \cap V_{\tilde{r}}| \geq \frac{49}{50} \cdot \alpha n$ .

**Proof of iii).** Note that we are proving this bound holds even if we remove the termination condition from the algorithm. This will show that we will reach round  $\tilde{r}$  with high probability. We first condition on the events in the proofs of *i*) and *ii*). Note that, unlike the standard analysis of elimination-based algorithms, here, we cannot directly upper-bound the total number of queries each round. Instead, we separately analyze the number of queries induced by the vertices in  $I^*$  and  $V \setminus I^*$ .

We first analyze the number of queries induced by the vertices in  $V \setminus I^*$ . Let us define  $X_{-I^*}$  as the total number of queries induced by the non-MIS vertices. Similarly, we can define  $X_{-I^*}^r$  as the queries induced by the non-MIS vertices at round  $r$ . Thus, we have that

$$\begin{aligned} \mathbb{E}[X_{-I^*}] &= \sum_{v \in V - I^*} \sum_{i=1}^{\tilde{r}} \Pr(v \text{ survives till round } i) \cdot q_i \\ &\leq n \sum_{i=1}^{\tilde{r}} q_i \prod_{j=1}^i \Pr(v \text{ survives round } j \mid v \text{ survives till round } j-1) \\ &\leq n \sum_{i=1}^{\tilde{r}} q_i \prod_{j=1}^i \frac{\delta}{100} \cdot \frac{1}{4^j} && \text{(Claim 6)} \\ &\leq n \sum_{i=1}^{\tilde{r}} \left(\frac{\delta}{100}\right)^i \cdot \left(\frac{1}{4}\right)^{\binom{i}{2}} \cdot q_i \\ &= n \sum_{i=1}^{\tilde{r}} \left(\frac{\delta}{100}\right)^i \cdot \left(\frac{1}{4}\right)^{\binom{i}{2}} \cdot \frac{4}{\varepsilon^2} \cdot (i + \log 1/\delta) \\ &\leq \frac{4\delta n}{100\varepsilon^2} \sum_{i=1}^{\tilde{r}} \left(\frac{1}{4}\right)^i \cdot (i + \log 1/\delta) && \text{(Since } \delta \leq 1) \\ &\leq \frac{4\delta n}{100\varepsilon^2} (1 + \log 1/\delta). && \text{(using properties of geometric sums)} \end{aligned}$$

Therefore, by Markov inequality, we can show that

$$\Pr\left(X_{-I^*} > \frac{2n}{5\varepsilon^2} \log 1/\delta\right) \leq \delta/5.$$

We now analyze the queries induced by the vertices in  $I^*$ . Similar to the case of the non-MIS analysis, let us define  $X_{I^*}$  as the total number of queries induced by the MIS vertices. We will trivially upper bound  $X_{I^*}$  in the following way:

$$\begin{aligned} X_{I^*} &\leq \alpha n \sum_{i=1}^{\tilde{r}} q_i \\ &= \alpha n \sum_{i=1}^{\tilde{r}} \frac{4}{\varepsilon^2} \cdot \left(i + \log \frac{1}{\delta}\right) \end{aligned}$$

## 24:12 Learning-Augmented Maximum Independent Set

$$\begin{aligned}
&\leq \frac{4\alpha n}{\varepsilon^2} \cdot \left( \tilde{r}^2 + \tilde{r} \cdot \log \frac{1}{\delta} \right) \\
&\leq \frac{4\alpha n}{\varepsilon^2} \cdot \left( 1 + (\log 1/\alpha)^2 + \lg 1/\alpha \cdot (2 + \log 1/\delta) + \log 1/\delta \right) \\
&\leq \frac{4n}{\varepsilon^2} \cdot (5 + 2 \log 1/\delta) \quad (\text{using } \alpha \cdot \lg \frac{1}{\alpha} \leq 1 \text{ and } \alpha \cdot \lg^2 \frac{1}{\alpha} \leq 2 \text{ for any } \alpha \in (0, 1))
\end{aligned}$$

We can then add the number of queries used by  $X_{\neg I^*}$  and  $X_{I^*}$  to get the desired sample complexity bound of  $\frac{30n}{\varepsilon^2} \cdot \log 1/\delta$ .

Finally, we can apply a union bound over the failure probabilities of the events in the proofs of *i*), *ii*), and *iii*) to argue that with probability at least  $1 - \delta$ , all the statements hold.

Lemma 7 ◀

We now proceed to show the guarantee of the matching and MIS phase. Our main lemma for this part is as follows.

► **Lemma 8.** *Let  $V_r \subseteq V$  be any subset of vertices in Algorithm 2. Furthermore, assume that the number of MIS vertices in  $V_r$  is at least 50 times the number of non-MIS vertices in  $V_r$ , i.e.,*

$$|V_r \cap I^*| \geq 50 \cdot |V_r \setminus I^*|.$$

Then, the set  $I_r$  returned by Algorithm 2 is a valid independent set, and we have

$$|I_r| \geq \frac{49}{50} \cdot |V_r \cap I^*|.$$

**Proof.** Recall that we compute a 2-approximate vertex cover  $U_r$  in the vertex cover phase. We know that the complement  $I_r \leftarrow V_r \setminus U_r$  is an independent set. This is because all edges of the graph are incident on the vertex cover so the remaining vertices form an independent set.

We know that  $V_r \setminus I^*$  is a vertex cover since  $V_r \cap I^*$  is an independent set. Thus, we have

$$\begin{aligned}
|I_r| &= |V_r| - |U_r| && (\text{by definition}) \\
&\geq |V_r \cap I^*| + |V_r \setminus I^*| - 2|V_r \setminus I^*| && (\text{since } U_r \text{ is a 2-approximation}) \\
&\geq |V_r \cap I^*| - \frac{1}{50} \cdot |V_r \cap I^*| && (\text{using the assumption}) \\
&= \frac{49}{50} \cdot |V_r \cap I^*|
\end{aligned}$$

Lemma 8 ◀

The final missing piece is the *efficiency* of the algorithm. We now prove that the algorithm is efficient both in time and the number of  $\text{ORC}_{G, I^*}$  oracle queries.

► **Lemma 9.** *Algorithm 2 runs in  $O(m \log n)$  time and uses at most  $\frac{30n}{\varepsilon^2} \cdot \log \frac{1}{\delta}$  queries on  $\text{ORC}_{G, I^*}$ .*

**Proof.** The query complexity is by the design of the algorithm as we terminate upon using more than  $30 \cdot \frac{n}{\varepsilon^2} \cdot \log \frac{1}{\delta}$  queries.

For the running time, note that in each iteration of  $r$ , we only need to: *i*). take the majority for all queried vertices, which can be maintained in  $O(n)$  time; and *ii*). compute a greedy matching and remove the vertices, which takes  $O(m)$  time. By Lemma 7, the process terminates in  $O(\log \frac{1}{\alpha}) = O(\log n)$  time ( $\alpha \geq \frac{1}{n}$  since there has to be at least one vertex in  $I^*$ ). Therefore, the entire algorithm takes  $O(m \log n)$  time in total. ◀

**Finalizing the proof of Theorem 5.** The query efficiency is by the design of the algorithm, and the running time simply follows from Lemma 9. For the approximation guarantee, note that by Lemma 7, we will proceed to round  $\tilde{r} = 10 \log \frac{1}{\alpha}$ , at which point we will have  $|V_{\tilde{r}} \cap I^*| \geq \frac{49}{50} \cdot \alpha n$  and  $|V_r \cap I^*| \geq 50 \cdot |V_r \setminus I^*|$ . Therefore, by Lemma 8, the returned  $I_{\tilde{r}}$  is of size at least

$$|I_{\tilde{r}}| \geq \frac{49}{50} \cdot |V_{\tilde{r}} \cap I^*| \geq \frac{49}{50} \cdot \frac{49}{50} \cdot \alpha n,$$

which gives us the desired  $48/50$  approximation.  $\blacktriangleleft$

► **Remark 10.** We aim to get the  $O(1)$  approximation in our algorithm and analysis. However, we remark that we can get both non-asymptotic and asymptotic trade-offs between the number of queries and the approximation factor. For the non-asymptotic trade-off (i.e., using more queries to get a better constant approximation), we can increase the leading constant on the sample complexity, and obtain the approximation with a larger constant. For the asymptotic trade-off, we can perform the simple trick by sampling  $k$  vertices uniformly at random and running Algorithm 2 on the sampled vertices. This will give us an  $O(\frac{k}{n})$ -approximation algorithm with  $O(\frac{k}{\varepsilon^2} \cdot \log \frac{1}{\delta})$  queries as long as  $\alpha k = \Omega(\log n)$ .

## 5 Discussion and Open Problems

We discussed learning-augmented algorithms for the Maximum Independent Set problem in this paper. Our main results include algorithms for both persistent and non-persistent noise settings, demonstrating that a learning-augmented oracle could lead to MIS algorithms with considerably better efficiency. There are several intriguing open problems following our work.

- For the **persistent noise** setting, the main open question is whether we could beat the  $\tilde{\Theta}(\sqrt{\Delta}/\varepsilon)$  approximation bound with the same oracle. We do not have any lower bounds for the persistent noise setting in this paper, and it is unclear what type of techniques could be used to prove lower bounds for learning-augmented algorithms.
- For the **non-persistent noise** setting, our algorithm matches the *asymptotically* optimal approximation factor using  $O(n)$  queries. In Appendix C of full version, we also proved that we cannot obtain the same results by only querying the oracle (and *not* looking into the graph). An open problem here is that if we want to recover a  $1 - o(1)$  fraction of the MIS vertices, how many queries do we need? We suspect there is a lower bound on the number of queries (e.g.,  $\omega(n)$ ), but it is not immediately clear how to prove it.
- We can also ask about **sublinear** number of queries on the oracle  $\text{ORC}_{G, I^*}$ , i.e., if we make  $o(n)$  queries on the oracle, what is the best we can do for both persistent and non-persistent noise settings? Currently, our algorithms in both settings require  $\Omega(n)$  queries to the oracle.
- Finally, for the **practical** aspect of the algorithms, we believe the oracles are possible to implement in practice. For instance, if we have features on the nodes, it is possible to use forward-pass graph convolution networks (GCNs), and simply run greedy in each “cluster” of nodes whose final features are sufficiently similar. Exploring practical oracles for this purpose would also be an interesting problem to resolve.

---

## References

- 1 Anders Aamand, Justin Y. Chen, Huy Lê Nguyen, Sandeep Silwal, and Ali Vakilian. Improved frequency estimation algorithms with and without predictions. *CoRR*, abs/2312.07535, 2023. doi:10.48550/arXiv.2312.07535.

- 2 Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 134–144. PMLR, 2020. URL: <http://proceedings.mlr.press/v119/ahn20a.html>.
- 3 Vladimir E. Alekseev, Vadim V. Lozin, Dmitriy S. Malyshev, and Martin Milanic. The maximum independent set problem in planar graphs. In Edward Ochmanski and Jerzy Tyszkiewicz, editors, *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, volume 5162 of *Lecture Notes in Computer Science*, pages 96–107. Springer, 2008. doi:10.1007/978-3-540-85238-4\_7.
- 4 Diogo V Andrade, Mauricio GC Resende, and Renato F Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18:525–547, 2012.
- 5 Antonios Antoniadis, Hajo Broersma, and Yang Meng. Online graph coloring with predictions. *arXiv preprint*, 2023. arXiv:2312.00601.
- 6 Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Discrete-smoothness in online algorithms with predictions. *Advances in Neural Information Processing Systems*, 36, 2024.
- 7 Eric Balkanski, Vasilis Gkatzelis, Xizhi Tan, and Cherlin Zhu. Online mechanism design with predictions. *CoRR*, abs/2310.02879, 2023. doi:10.48550/arXiv.2310.02879.
- 8 Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020.
- 9 Siddhartha Banerjee, Vincent Cohen-Addad, Anupam Gupta, and Zhouzi Li. Graph searching with predictions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPICs*, pages 12:1–12:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ITCS.2023.12.
- 10 Ravi Boppana and Magnús M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, 1992.
- 11 Ravi B Boppana. Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 280–285. IEEE, 1987.
- 12 Nicolas Bourgeois, Bruno Escoffier, Vangelis Th. Paschos, and Johan M. M. van Rooij. Fast algorithms for max independent set. *Algorithmica*, 62(1-2):382–415, 2012. doi:10.1007/S00453-010-9460-7.
- 13 Jan van den Brand, Sebastian Forster, Yasamin Nazari, and Adam Polak. On dynamic graph algorithms with predictions. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3534–3557. SIAM, 2024.
- 14 Lorenzo Brusca, Lars C. P. M. Quaedvlieg, Stratis Skoulakis, Grigorios Chrysos, and Volkan Cevher. Maximum independent set: Self-training through dynamic programming. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL: [http://papers.nips.cc/paper\\_files/paper/2023/hash/7fe3170d88a8310ca86df2843f54236c-Abstract-Conference.html](http://papers.nips.cc/paper_files/paper/2023/hash/7fe3170d88a8310ca86df2843f54236c-Abstract-Conference.html).
- 15 Wei Cao, Jian Li, Yufei Tao, and Zhize Li. On top-k selection in multi-armed bandits and hidden bipartite graphs. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1036–1044, 2015. URL: <https://proceedings.neurips.cc/paper/2015/hash/ab233b682ec355648e7891e66c54191b-Abstract.html>.
- 16 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 892–901. SIAM, 2009. doi:10.1137/1.9781611973068.97.



- 17 Justin Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning*, pages 3583–3602. PMLR, 2022.
- 18 Justin Y Chen, Talya Eden, Piotr Indyk, Honghao Lin, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, Tal Wagner, David P Woodruff, and Michael Zhang. Triangle and four cycle counting with predictions in graph streams. *arXiv preprint*, 2022. [arXiv:2203.09572](https://arxiv.org/abs/2203.09572).
- 19 Lijie Chen, Jian Li, and Mingda Qiao. Nearly instance optimal sample complexity bounds for top-k arm selection. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 101–110. PMLR, 2017. URL: <http://proceedings.mlr.press/v54/chen17a.html>.
- 20 Jakub Chledowski, Adam Polak, Bartosz Szabucki, and Konrad Tomasz Zolna. Robust learning-augmented caching: An experimental study. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1920–1930. PMLR, 2021. URL: <http://proceedings.mlr.press/v139/chledowski21a.html>.
- 21 Nicolas Christianson, Tinashe Handina, and Adam Wierman. Chasing convex bodies and functions with black-box advice. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 867–908. PMLR, 2022. URL: <https://proceedings.mlr.press/v178/christianson22a.html>.
- 22 Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. 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 820–829. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.92.
- 23 Vincent Cohen-Addad, Tommaso d’Orsi, Anupam Gupta, Euiwoong Lee, and Debmalya Panigrahi. Max-cut with  $\epsilon$ -accurate predictions. *CoRR*, abs/2402.18263, 2024. doi:10.48550/arXiv.2402.18263.
- 24 Jakob Dahlum, Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. Accelerating local search for the maximum independent set problem. In Andrew V. Goldberg and Alexander S. Kulikov, editors, *Experimental Algorithms - 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings*, volume 9685 of *Lecture Notes in Computer Science*, pages 118–133. Springer, 2016. doi:10.1007/978-3-319-38851-9\_9.
- 25 Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.
- 26 Jon C. Ergun, Zhili Feng, Sandeep Silwal, David P. Woodruff, and Samson Zhou. Learning-augmented  $k$ -means clustering. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: <https://openreview.net/forum?id=X8cLTHexYyY>.
- 27 Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and markov decision processes. In Jyrki Kivinen and Robert H. Sloan, editors, *Computational Learning Theory, 15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 8-10, 2002, Proceedings*, volume 2375 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2002. doi:10.1007/3-540-45435-7\_18.
- 28 Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *J. Mach. Learn. Res.*, 7:1079–1105, 2006. URL: <http://jmlr.org/papers/v7/evendar06a.html>.
- 29 Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics*, 18(2):219–225, 2004.
- 30 Willem Feijen and Guido Schäfer. Using machine learning predictions to speed-up dijkstra’s shortest path algorithm. *CoRR*, pages 1–28, 2021.

- 31 Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Measure and conquer: a simple  $o(2^{0.288n})$  independent set algorithm. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 18–25. ACM Press, 2006. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109560>.
- 32 Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A 3-approximation algorithm for maximum independent set of rectangles. 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 894–905. SIAM, 2022. doi:10.1137/1.9781611977073.38.
- 33 Suprovat Ghoshal, Konstantin Makarychev, and Yury Makarychev. Constraint satisfaction problems with advice. *arXiv preprint*, 2024. arXiv:2403.02212.
- 34 Francesco Gullo, Domenico Mandaglio, and Andrea Tagarelli. A combinatorial multi-armed bandit approach to correlation clustering. *Data Min. Knowl. Discov.*, 37(4):1630–1691, 2023. doi:10.1007/S10618-023-00937-5.
- 35 Shubham Gupta, Peter W. J. Staar, and Christian de Sainte Marie. Clustering items from adaptively collected inconsistent feedback. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pages 604–612. PMLR, 2024. URL: <https://proceedings.mlr.press/v238/gupta24a.html>.
- 36 Johan Håstad. Clique is hard to approximate within  $n^{(1-\epsilon)}$ . In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636. IEEE Computer Society, 1996. doi:10.1109/SFCS.1996.548522.
- 37 Monika Henzinger, Andrea Lincoln, Barna Saha, Martin P Seybold, and Christopher Ye. On the complexity of algorithms with predictions for dynamic graph problems. *arXiv preprint*, 2023. arXiv:2307.16771.
- 38 Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: <https://openreview.net/forum?id=r11ohoCqY7>.
- 39 Bingbing Hu, Evangelos Kosinas, and Adam Polak. Connectivity oracles for predictable vertex failures. *arXiv preprint*, 2023. arXiv:2312.08489.
- 40 Shivaram Kalyan Krishnan and Peter Stone. Efficient selection of multiple bandit arms: Theory and practice. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 511–518. Omnipress, 2010. URL: <https://icml.cc/Conferences/2010/papers/410.pdf>.
- 41 Shivaram Kalyan Krishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL: <http://icml.cc/2012/papers/359.pdf>.
- 42 Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.
- 43 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.
- 44 Yuko Kuroki, Atsushi Miyauchi, Francesco Bonchi, and Wei Chen. Query-efficient correlation clustering with noisy oracle. *CoRR*, abs/2402.01400, 2024. doi:10.48550/arXiv.2402.01400.
- 45 Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. Speeding up bellman ford via minimum violation permutations. In *International Conference on Machine Learning*, pages 18584–18598. PMLR, 2023.

- 46 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPICs*, pages 59:1–59:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ESA.2021.59.
- 47 Avner Magen and Mohammad Moharrami. Robust algorithms for MAX INDEPENDENT SET on minor-free graphs based on the sherali-adams hierarchy. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, volume 5687 of *Lecture Notes in Computer Science*, pages 258–271. Springer, 2009. doi:10.1007/978-3-642-03685-9\_20.
- 48 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.
- 49 Thomas Pontoizeau, Florian Sikora, Florian Yger, and Tristan Cazenave. Neural maximum independent set. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I*, volume 1524 of *Communications in Computer and Information Science*, pages 223–237. Springer, 2021. doi:10.1007/978-3-030-93736-2\_18.
- 50 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. 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 9684–9693, 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html>.
- 51 John M Robson. Finding a maximum independent set in time  $o(2n/4)$ . Technical report, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.
- 52 Tim Roughgarden. *Beyond the worst-case analysis of algorithms*. Cambridge University Press, 2021.
- 53 Karim Abdel Sadek and Marek Elias. Algorithms for caching and mts with reduced number of predictions. *arXiv preprint*, 2024. arXiv:2404.06280.
- 54 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.
- 55 Max Simchowitz, Kevin G. Jamieson, and Benjamin Recht. The simulator: Understanding adaptive sampling in the moderate-confidence regime. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, pages 1794–1834. PMLR, 2017. URL: <http://proceedings.mlr.press/v65/simchowitz17a.html>.
- 56 Clifford Stein and Hao-Ting Wei. Learning-augmented online packet scheduling with deadlines. *CoRR*, abs/2305.07164, 2023. doi:10.48550/arXiv.2305.07164.
- 57 Jose L. Walteros and Austin Buchanan. Why is maximum clique often easy in practice? *Oper. Res.*, 68(6):1866–1895, 2020. doi:10.1287/OPRE.2019.1970.
- 58 Jinghui Xia and Zengfeng Huang. Optimal clustering with noisy queries via multi-armed bandit. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 24315–24331. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/xia22a.html>.
- 59 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. In Leizhen Cai, Siu-Wing Cheng, and Tak Wah Lam, editors, *Algorithms and Computation - 24th International Symposium, ISAAC 2013, Hong Kong, China, December 16-18, 2013, Proceedings*, volume 8283 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2013. doi:10.1007/978-3-642-45030-3\_31.

- 60 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006.

## A Technical Preliminaries

We use the following standard forms of Chernoff bound.

► **Proposition 11** (Chernoff-Hoeffding bound). *Let  $X_1, \dots, X_m$  be  $m$  independent random variables with support in  $[0, 1]$ . Define  $X := \sum_{i=1}^m X_i$ . Then, for every  $t > 0$ ,*

$$\Pr(X - \mathbb{E}[X] \geq t) \leq \exp\left(-\frac{2t^2}{m}\right)$$

$$\Pr(X - \mathbb{E}[X] \leq -t) \leq \exp\left(-\frac{2t^2}{m}\right).$$

► **Proposition 12** (Chernoff bound; c.f. [25]). *Suppose  $X_1, \dots, X_m$  are  $m$  independent random variables with range  $[0, 1]$  each. Let  $X := \sum_{i=1}^m X_i$  and  $\mu_L \leq \mathbb{E}[X] \leq \mu_H$ . Then, for any  $\delta \in [0, 1]$ ,*

$$\Pr(X > (1 + \delta) \cdot \mu_H) \leq \exp\left(-\frac{\delta^2 \cdot \mu_H}{3 + \delta}\right) \quad \text{and} \quad \Pr(X < (1 - \delta) \cdot \mu_L) \leq \exp\left(-\frac{\delta^2 \cdot \mu_L}{2 + \delta}\right).$$

We also consider limited independence hash functions. Roughly speaking, a  $k$ -wise independent hash function behaves like a totally random function when considering at most  $k$  elements. Formally, a family of hash functions  $H = \{h : [n] \rightarrow [m]\}$  is  $k$ -wise independent if for any  $x_1, x_2, \dots, x_k \in [n]$  and  $y_1, y_2, \dots, y_k \in [m]$  the following holds:

$$\Pr_{h \in_R H}(h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \dots \wedge h(x_k) = y_k) = m^{-k}.$$

We shall use the following concentration result on an extension of Chernoff-Hoeffding bounds for limited independence hash function.

► **Proposition 13** ([54]). *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  $\delta > 0$ ,*

$$\Pr(|X - \mathbb{E}[X]| \geq \delta \cdot \mathbb{E}[X]) \leq \exp\left(-\min\left\{\frac{k}{2}, \frac{\delta^2}{4 + 2\delta} \cdot \mathbb{E}[X]\right\}\right).$$

# Maximum Unique Coverage on Streams: Improved FPT Approximation Scheme and Tighter Space Lower Bound

Philip Cervenjak ✉ 

School of Computing and Information Systems, The University of Melbourne, Australia

Junhao Gan ✉ 

School of Computing and Information Systems, The University of Melbourne, Australia

Seeun William Umboh ✉ 

School of Computing and Information Systems, The University of Melbourne, Australia

ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Melbourne, Australia

Anthony Wirth ✉ 

School of Computer Science, The University of Sydney, Australia

School of Computing and Information Systems, The University of Melbourne, Australia

---

## Abstract

We consider the Max Unique Coverage problem, including applications to the data stream model. The input is a universe of  $n$  elements, a collection of  $m$  subsets of this universe, and a cardinality constraint,  $k$ . The goal is to select a subcollection of at most  $k$  sets that maximizes unique coverage, i.e., the number of elements contained in exactly one of the selected sets. The Max Unique Coverage problem has applications in wireless networks, radio broadcast, and envy-free pricing.

Our first main result is a fixed-parameter tractable approximation scheme (FPT-AS) for Max Unique Coverage, parameterized by  $k$  and the maximum element frequency,  $r$ , which can be implemented on a data stream. Our FPT-AS finds a  $(1 - \varepsilon)$ -approximation while maintaining a kernel of size  $\tilde{O}(kr/\varepsilon)$ , which can be combined with subsampling to use  $\tilde{O}(k^2r/\varepsilon^3)$  space overall. This significantly improves on the previous-best FPT-AS with the same approximation, but a kernel of size  $\tilde{O}(k^2r/\varepsilon^2)$ . In order to achieve our first result, we show upper bounds on the ratio of a collection's coverage to the unique coverage of a maximizing subcollection; this is by constructing explicit algorithms that find a subcollection with unique coverage at least a logarithmic ratio of the collection's coverage. We complement our algorithms with our second main result, showing that  $\Omega(m/k^2)$  space is necessary to achieve a  $(1.5 + o(1))/(\ln k - 1)$ -approximation in the data stream. This dramatically improves the previous-best lower bound showing that  $\Omega(m/k^2)$  is necessary to achieve better than a  $e^{-1+1/k}$ -approximation.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Maximum unique coverage, maximum coverage, approximate kernel, data streams

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.25

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.09368>

**Funding** *Philip Cervenjak:* This work was supported by the Elizabeth and Vernon Puzey Scholarship, and by the Faculty of Engineering and Information Technology.

**Acknowledgements** We thank the anonymous reviewers for their valuable feedback.



© Philip Cervenjak, Junhao Gan, Seeun William Umboh, and Anthony Wirth; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 25; pp. 25:1–25:23



Leibniz International Proceedings in Informatics

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

## 1 Introduction

We study the **Max Unique Coverage** problem, where we are given a universe of  $n$  elements, a collection of  $m$  subsets of the universe, and an integer  $k \in \{1, \dots, m\}$ . The goal is to select a collection of at most  $k$  subsets that maximizes the number of elements covered by *exactly* one set in the collection. This problem is a natural variant of the classic **Max Coverage** problem, where the goal is to select a collection of  $k$  subsets that maximizes the number of elements covered by *at least* one set in the collection.

A weighted version of **Max Unique Coverage** was first formally studied by Demaine et al. [9]. In their motivating scenario, a number of wireless base stations, each with an associated cost, must be placed to maximize the number of mobile clients served. However, due to interference, if covered by more than one base station, a client receives bad service. Demaine et al. point out further applications to radio broadcast and envy-free pricing. They then showed an offline polynomial-time  $\Omega(1/\log m)$ -approximation algorithm for their problem, which easily translates to a  $\Omega(1/\log k)$ -approximation for our problem.<sup>1</sup> Under various complexity assumptions, they showed (semi-)logarithmic inapproximability for polynomial-time algorithms; Guruswami and Lee [11] later proved nearly logarithmic inapproximability, assuming NP does not admit quasipolynomial-time algorithms.

**Streaming.** Our work emphasizes solving **Max Unique Coverage** approximately in the data stream model. All previous works, except McGregor et al. [15], only consider this problem in the offline model. In the data stream model, we focus on *set-streaming*: each set in the stream is fully specified before the next; this setting is assumed in related works [18, 2, 22, 16, 15]. We also constrain the space, measured in bits, to be  $o(mn)$ , i.e., sublinear in both the number of sets,  $m$ , and the size of the universe,  $n$ . Thus, we define the **Max Unique Coverage** problem to include the cardinality constraint,  $k$ . Previous works often formulate this problem without a cardinality constraint, simply referring to it as the “Unique Coverage” problem; this is equivalent to our formulation when  $k = m$ .

We are particularly interested in **Max Unique Coverage** when parameterized by  $k$  and the *maximum frequency*,  $r$ , defined as the maximum number of sets that an element belongs to; we also consider the *maximum set size*  $d$  to a lesser extent. This parameterization has received considerable attention in studying fixed-parameter tractable approximation schemes (FPT-AS) for classic coverage problems [21, 4, 20, 14, 15, 13, 19], but not as much for the **Max Unique Coverage** problem [15]. Note that an FPT exact algorithm for this parameterization is unlikely to exist because, when  $r = 2$ , **Max Unique Coverage** is equivalent to Capacitated Max Cut, which was shown by Misra et al. [17] to be W[1]-hard when parameterized by the capacity constraint.<sup>2</sup>

A central idea in achieving both FPT space and running time bounds is *kernelization*. We transform a problem instance,  $\mathcal{I}$ , into a smaller problem instance,  $\mathcal{I}'$ , called the (*approximate*) *kernel*, such that  $|\mathcal{I}'| \leq g(\gamma)$ , where  $g$  is a computable function in terms of problem parameters  $\gamma$ , while  $\mathcal{I}'$  (approximately) preserves the optimal solution value of  $\mathcal{I}$ ; a good solution can be found by brute-force search within  $\mathcal{I}'$ . Consistent with other works [8, 7, 6, 15], we further require an FPT streaming algorithm to use  $g(\gamma)$  polylog  $|\mathcal{I}|$  space.

<sup>1</sup> This is by assuming that all sets have unit cost and that the budget is  $k$ .

<sup>2</sup> Although Misra et al. [17] prove W[1]-hardness for Budgeted Max Cut when parameterized by the budget, their hardness proof only requires each vertex (corresponding to a set in our formulation) to have unit cost.

## 1.1 Our Contributions

Our first main result is an FPT-AS for **Max Unique Coverage** with strong space, running time, and approximation bounds, that is applicable to the data stream model. A crucial step in achieving such bounds is showing improved upper bounds on a key parameter of a collection  $\mathcal{C}$ . The *unique coverage ratio* is the ratio between the coverage of  $\mathcal{C}$  and the maximum unique coverage over all subcollections  $\mathcal{Q} \subseteq \mathcal{C}$ . We let  $\phi$  denote an *upper bound* on the unique coverage ratio: we define performance bounds of our FPT-AS in terms of  $\phi$ .

**Main Result 1: FPT Approximation Scheme.** We propose the FPT-AS `UNIQUETOPSETS`, parameterized by the cardinality constraint,  $k$ , and the maximum frequency,  $r$ , which can be easily implemented in the data stream model. It achieves a  $(1 - \varepsilon)$ -approximation using a kernel of size  $\lceil kr(\phi + 1)/\varepsilon \rceil$ . We formally present this algorithm in Theorem 3.6.

`UNIQUETOPSETS` is a refined version of the FPT-AS in Theorem 12 of McGregor et al. [15], in that our algorithm achieves a  $(1 - \varepsilon)$  rather than a  $(1/2 - \varepsilon)$ -approximation using only an extra logarithmic factor of  $(\phi + 1)$  in the kernel size. Further, our algorithm improves on the FPT-AS in Theorem 10 of McGregor et al. [15] by saving a factor of  $O(k/\varepsilon)$  in the kernel size, and therefore a factor of  $[O(k/\varepsilon)]^k$  in the running time, while achieving the same approximation factor. See Table 1 for a comparison of our FPT-AS with others.

■ **Table 1** Comparison of FPT-AS for **Max Unique Coverage**, parameterized by cardinality constraint,  $k$ , and maximum frequency,  $r$ . Note that the running time of each algorithm below is implied by its kernel size. Each finds a solution of size at most  $k$  by brute-force search in the kernel. Below, we can assign  $\phi = \min(\ln k + 1, 2 \ln r + o(\log r), 2 \ln d + o(\log d))$ .

| Reference  | Approx.             | Kernel Size                              |
|--|---------------------|--|
| [15, Theorem 10]                                 | $1 - \varepsilon$   | $O(k^2 r \log m / \varepsilon^2)$        |
| [15, Theorem 12]                                 | $1/2 - \varepsilon$ | $\lceil kr/\varepsilon \rceil$           |
| Ours, Theorem 3.6 ( <code>UNIQUETOPSETS</code> ) | $1 - \varepsilon$   | $\lceil kr(\phi + 1)/\varepsilon \rceil$ |

**Unique Coverage Algorithms.** In order to show good values for  $\phi$ , we propose a number of offline polynomial-time algorithms that, given an arbitrary  $\mathcal{C}$ , explicitly return a  $\mathcal{B} \subseteq \mathcal{C}$  whose unique coverage is at least a logarithmic ratio of  $\mathcal{C}$ 's coverage. We refer to them as *unique coverage algorithms*; in fact, they can be thought as approximation algorithms for the unconstrained Unique Coverage problem on an input instance of  $\mathcal{C}$ .

Our three offline polynomial-time algorithms, `UNIQUEGREEDY`, `UNIQUEGREEDYFREQ`, and `UNIQUEGREEDYSIZE`, each take a collection of sets,  $\mathcal{C}$ , and return a collection,  $\mathcal{B} \subseteq \mathcal{C}$ , whose unique coverage is at least a  $1/(\ln \ell + 1)$ ,  $1/(2 \ln r + o(\log r))$ , and  $1/(2 \ln d + o(\log d))$  proportion of  $\mathcal{C}$ 's coverage respectively; in this context,  $\ell = |\mathcal{C}|$ ,  $r$  is the maximum frequency in  $\mathcal{C}$ , and  $d$  is the maximum set size in  $\mathcal{C}$ . We formally present these algorithms in Theorem 4.1, Theorem 4.4, and in Theorem 4.8, respectively. See Table 2 for a comparison of our algorithms with those of Demaine et al. [9] along with their implied bounds,  $\phi$ , albeit weaker than ours.

■ **Table 2** Polynomial-time algorithms for **Max Unique Coverage**. Compared to others, our methods imply constant-factor improvements in the unique coverage ratio bound,  $\phi$ .

| Parameter                                  | Reference                               | (Implied) $\phi$      |
|--|---|-----------------------|
| $\ell =$ collection size                   | [9, Theorem 4.1]                        | $10.66 \ln(\ell + 1)$ |
|  | Ours, Theorem 4.1<br>(UNIQUEGREEDY)     | $\ln \ell + 1$        |
| $r =$ maximum frequency<br>in a collection | [9, Theorem 4.1]                        | $10.66 \ln(r + 1)$    |
|  | Ours, Theorem 4.4<br>(UNIQUEGREEDYFREQ) | $2 \ln r + o(\log r)$ |
| $d =$ maximum set size<br>in a collection  | [9, Theorem 4.2]                        | $21.32 \ln(d + 1)$    |
|  | Ours, Theorem 4.8<br>(UNIQUEGREEDYSIZE) | $2 \ln d + o(\log d)$ |

**Implication for FPT Approximation Scheme.** The bound on the unique coverage ratio,  $\phi$ , affects the kernel size and therefore the brute-force running time of UNIQUETOPSETS. In particular, when  $r = \Omega(\sqrt{k})$ , the bound of  $\phi$  implied by UNIQUEGREEDY is 10.66 times smaller than implied by Demaine et al. [9]; whereas when  $r = o(\sqrt{k})$ , the bound of  $\phi$  implied by UNIQUEGREEDYFREQ is almost 5.33 smaller than implied by Demaine et al. This means, by using our implied bounds rather than those implied by Demaine et al., we save a factor of  $10.66^k$  in UNIQUETOPSETS’s running-time when  $r = \Omega(\sqrt{k})$ , and a factor of almost  $5.33^k$  when  $r = o(\sqrt{k})$ .

**Improvements in Polynomial-Time Approximation.** As a separate contribution, each of our three unique coverage algorithms finds a logarithmic approximation to **Max Unique Coverage**, both offline and in the data stream. We first find a solution  $\mathcal{C}$  to **Max Coverage** in polynomial time, and then run one of our above algorithms on  $\mathcal{C}$  to return the subcollection  $\mathcal{B} \subseteq \mathcal{C}$ . For this purpose, our algorithms UNIQUEGREEDY, UNIQUEGREEDYFREQ, and UNIQUEGREEDYSIZE improve the approximation factor due to Demaine et al. [9] by a factor of 10.66, 5.33, and 10.66, respectively. Following the above approach, we propose a single-pass streaming algorithm for **Max Unique Coverage** that achieves a  $(1/(2\phi) - \varepsilon)$ -approximation using  $\tilde{O}(k^2/\varepsilon^3)$  space, where we can assign  $\phi = \min(\ln k + 1, 2 \ln r + o(\log r), 2 \ln d + o(\log d))$ . We formally state this in Theorem 3.8.

**Main Result 2: Streaming Lower Bound.** Our second main result is a significantly improved streaming lower bound for **Max Unique Coverage**. In the data stream model, we prove that any randomized algorithm that achieves a  $(1.5 + o(1))/(\ln k - 1)$ -approximation for **Max Unique Coverage** w.h.p. requires  $\Omega(m/k^2)$  space. We formally state this in Theorem 5.1. Our lower bound improves on the lower bound by McGregor et al. [15], which shows a similar result, but achieves w.h.p. a  $e^{-1+1/k} \geq 1/e$ -approximation. Interestingly, our approximation threshold is close to 3 times larger than the approximation (in terms of  $k$ ) achieved by our  $\tilde{O}(k^2/\varepsilon^3)$  space algorithm in Theorem 3.8, indicating that a dramatic increase in space is needed to bridge this approximation gap.



■ **Table 3** Comparison of space lower bounds for **Max Unique Coverage** in the data stream. Note that the lower bound by Assadi [1] was shown for **Max Coverage** with constant  $k = 2$ , but it is not difficult to adapt it for **Max Unique Coverage** because, in the hard instance constructed for the lower bound, the unique coverage of any pair of sets behaves similarly to its coverage.

| Reference         | Approx.                    | Space LB                  |
|-------------------|----------------------------|---------------------------|
| [1, Theorem 4]    | $1 - \varepsilon$          | $\Omega(m/\varepsilon^2)$ |
| [15, Theorem 16]  | $1/e$                      | $\Omega(m/k^2)$           |
| Ours, Theorem 5.1 | $(1.5 + o(1))/(\ln k - 1)$ | $\Omega(m/k^2)$           |

## 1.2 Technical Overview

**FPT Approximation Scheme.** `UNIQUETOPSETS` refines the technique used in the FPT-AS for **Max Unique Coverage** in Theorem 12 of McGregor et al. [15], which is to construct an approximate kernel by storing a number of the largest sets by individual size, and then to find a subcollection of the kernel with maximum unique coverage by brute-force search. Similar techniques have been used in FPT-AS approaches for Max Vertex Cover [14, 13] and **Max Coverage** [21, 20, 15, 19]. Our novelty is providing a stronger analysis of the approximation factor preserved by the kernel, allowing us to achieve a  $(1 - \varepsilon)$ -approximation while only increasing the kernel size by a logarithmic factor in  $k$ ,  $r$ , or  $d$ .

**Unique Coverage Algorithms.** All of our unique coverage algorithms are combinatorial in design. Our first two, `UNIQUEGREEDY` and `UNIQUEGREEDYFREQ`, are novel algorithms that each, in some sense, use a greedy approach, noting that `UNIQUEGREEDY` is used as subroutine of `UNIQUEGREEDYFREQ`. Our third algorithm, `UNIQUEGREEDYSIZE`, is easily derived by combining `UNIQUEGREEDYFREQ` with the approach by Demaine et al. [9] for sets with maximum cost  $d$  (maximum size in our case).

**Streaming Lower Bound.** Our streaming lower bound relies on a novel reduction from  $k$ -player Set Disjointness in the one-way communication model to **Max Unique Coverage** in the data stream. In the hard instance of **Max Unique Coverage** thus constructed, either all collections of  $\ell \leq k$  sets have a unique coverage of  $ak^2(1.5 + o(1))$  w.h.p. or there exists a single collection of  $k$  sets whose unique coverage is at least  $ak^2(\ln k - 1)$ , where  $a = \Omega(k \log m)$ . By a standard argument, we show that distinguishing between these instances of **Max Unique Coverage** with a streaming algorithm is as hard as solving Set Disjointness, implying the required space lower bound.

## 1.3 Paper Structure

After preliminaries in Section 2, Section 3 presents our FPT-AS `UNIQUETOPSETS` and a polynomial-time algorithm, both applicable to the data stream. In Section 4, we present our component algorithms for bounding the unique coverage ratio. In Section 5, we present a space lower bound for achieving a  $(1.5 + o(1))/(\ln k - 1)$ -approximation for **Max Unique Coverage**. We conclude in Section 7. Claims whose proofs are found in the full version of this paper are marked thus: (\*).

## 2 Preliminaries

**Notation.** For convenience, we hence let  $[n]$  denote the set of integers  $\{1, 2, \dots, n\}$ . Likewise,  $U = [n]$  denotes a universe of  $n$  elements, while  $\mathcal{V}$  denotes a collection of  $m$  subsets of  $U$ .

Given a collection  $\mathcal{C}$  of sets, the *unique cover* of  $\mathcal{C}$  is the subset of the universe covered by exactly one set in  $\mathcal{C}$ , formally,  $\tilde{\psi}(\mathcal{C}) := (\bigcup_{S \in \mathcal{C}} S) \setminus (\bigcup_{S \neq T \in \mathcal{C}} S \cap T)$ , and the *unique coverage* of  $\mathcal{C}$  is  $|\tilde{\psi}(\mathcal{C})|$ . For convenience, the *cover* of  $\mathcal{C}$  is the union of the sets in  $\mathcal{C}$ , formally  $\psi(\mathcal{C}) := \bigcup_{S \in \mathcal{C}} S$ , and the *coverage* of  $\mathcal{C}$  is  $|\psi(\mathcal{C})|$ . Further, the *non-unique cover* of  $\mathcal{C}$  is the subset of the universe covered by at least two sets from  $\mathcal{C}$ , formally  $\psi_{\geq 2}(\mathcal{C}) = \bigcup_{S \neq T \in \mathcal{C}} S \cap T$  – or equivalently  $\psi_{\geq 2}(\mathcal{C}) = \psi(\mathcal{C}) \setminus \tilde{\psi}(\mathcal{C})$  – and the *non-unique coverage* of  $\mathcal{C}$  is  $|\psi_{\geq 2}(\mathcal{C})|$ . The *maximum unique coverage* of  $\mathcal{C}$  is the largest unique coverage of a subcollection of  $\mathcal{C}$ . The *unique coverage ratio* of  $\mathcal{C}$  is the ratio between its coverage and maximum unique coverage. In other words, if  $\mathcal{Q}$  is the subcollection of  $\mathcal{C}$  that has maximum unique coverage, then the unique coverage ratio of  $\mathcal{C}$  is  $|\psi(\mathcal{C})|/|\tilde{\psi}(\mathcal{Q})|$ .

Given an element  $x \in U$  and a collection  $\mathcal{C}$  of sets, the *frequency* of  $x$  in  $\mathcal{C}$  is defined as  $\text{freq}_{\mathcal{C}}(x) := |\{S \in \mathcal{C} : x \in S\}|$ , i.e., the number of sets in  $\mathcal{C}$  that contain  $x$ ; and the *maximum frequency* is defined as  $r := \max_{x \in U} \text{freq}_{\mathcal{C}}(x)$ . Also, the *maximum set size* is defined as  $d := \max_{S \in \mathcal{C}} |S|$ . We often use  $r$  and  $d$  to refer to the maximum frequency and set size, respectively, in  $\mathcal{C} = \mathcal{V}$  unless stated otherwise. Note that  $r \leq |\mathcal{C}|$  holds for every  $\mathcal{C}$ . We let  $H_z := \sum_{t=1}^z 1/t$  denote the  $z^{\text{th}}$  harmonic number, a term that appears several times.

**Formal Problem Definition.** An instance of **Max Unique Coverage** consists of an element universe  $U$ , a collection  $\mathcal{V}$  of  $m$  subsets of  $U$ , and an integer  $k \in [m]$ ; when the context is clear, we represent an instance with just  $\mathcal{V}$  for simplicity. The goal of **Max Unique Coverage** is to return a subcollection  $\mathcal{B} \subseteq \mathcal{V}$  (more precisely, a collection of IDs of sets), with  $|\mathcal{B}| \leq k$ , that maximizes  $|\tilde{\psi}(\mathcal{B})|$ . We let  $\mathcal{O}$  denote an optimal solution to this **Max Unique Coverage** problem, and  $\text{OPT} := |\tilde{\psi}(\mathcal{O})|$  as the maximum unique coverage.

**Subsampling for the Data Stream Model.** The universe subsampling technique has been widely successful in the development of streaming algorithms for coverage problems [10, 12, 3, 16]. In this work, we follow the approach of McGregor and Vu [16], and sample the universe so that each set has size  $O(k \log m / \varepsilon^2)$ . We assume that  $k \in o(mn)$ , and also that  $k$  is known prior to reading the stream. The main result is given in the following lemma, with a proof sketch of the subsampling approach in Section 6.

► **Lemma 2.1** (Subsampling Approach [15]). *Let  $\varepsilon \in (0, 1)$  be the subsampling error parameter. Given an instance of **Max Unique Coverage** and an  $\alpha$ -approximation streaming algorithm, we can run the algorithm on  $\lceil \log_2 n \rceil$  parallel subsampled instances and select one of them such that the algorithm's solution corresponds to a  $(\alpha - 2\varepsilon)$ -approximation for the original instance with probability  $1 - 1/\text{poly}(m)$ . Moreover, if the streaming algorithm stores at most  $s$  sets in every subsampled instance, then the total space complexity of the subsampling approach is bounded by  $\lceil \log_2 n \rceil \cdot s \cdot O(k \log m \log n / \varepsilon^2)$ .*

## 3 Streaming FPT-AS and Polynomial-Time Algorithms

In Section 3.1, we prove a kernelization lemma. Then, we use it to obtain an FPT-AS and a parameterized streaming algorithm in Section 3.2. Finally, we show how to use a bound on the unique coverage ratio to obtain a polynomial-time streaming algorithm in Section 3.3.

### 3.1 Kernelization Lemma

Our Kernelization Lemma below, as well as its proof, is a refinement of Lemma 11 by McGregor et al. [15]. We first provide some intuition on why our kernel preserves a  $(1 - \varepsilon)$ -approximation for **Max Unique Coverage**.

**Intuition of Kernelization Lemma.** For convenience, let  $\varepsilon'$  be an intermediate error parameter and define the kernel  $\mathcal{A}$  as the collection of the  $\lceil kr/\varepsilon' \rceil$  largest sets in instance  $\mathcal{V}$  by individual size. Given the optimal solution for **Max Unique Coverage**,  $\mathcal{O}$ , let  $\mathcal{O}^{\text{in}}$  and  $\mathcal{O}^{\text{out}}$  be the collections of optimal sets found and not found in  $\mathcal{A}$  respectively.

One main step in proving our Kernelization Lemma is showing that, in expectation, a collection of  $|\mathcal{O}^{\text{out}}|$  sets sampled without replacement from  $\mathcal{A}$ , denoted by  $\mathcal{Z}$ , can be appended to  $\mathcal{O}^{\text{in}}$  with little overlap in their unique covers. In particular, we can prove that  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|] \geq (1 - \varepsilon')|\tilde{\psi}(\mathcal{O})| - \varepsilon'|\psi(\mathcal{O})|$ .

However, due to the  $\varepsilon'|\psi(\mathcal{O})|$  term, this is not enough to achieve the required approximation factor. This term reflects the fact that, even if the unique cover of  $\mathcal{O}^{\text{in}}$  has little overlap with the unique cover of  $\mathcal{Z}$ , the *entire* cover of  $\mathcal{O}^{\text{in}}$  could be more extensive and, thus, overlap significantly with the unique cover of  $\mathcal{Z}$ . To address this, in Claim 3.4, we show  $\phi|\tilde{\psi}(\mathcal{O})| \geq |\psi(\mathcal{O})|$ , where  $\phi$  upper bounds the unique coverage ratio. Substituting this into the lower bound for  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|]$ , and assigning  $\varepsilon' = \varepsilon/(\phi + 1)$ , we obtain  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|] \geq (1 - \varepsilon)|\tilde{\psi}(\mathcal{O})|$ , implying the existence of a  $(1 - \varepsilon)$ -approximate subcollection of  $\mathcal{A}$ . Lastly, the final kernel size of  $|\mathcal{A}| = \lceil kr(\phi + 1)/\varepsilon \rceil$  follows from the assignment of  $\varepsilon'$ .

► **Lemma 3.1 (Kernelization Lemma).** *Suppose that every collection of sets has unique coverage ratio at most  $\phi$ . Let  $\mathcal{V}$  denote a collection of sets. Then, for every  $\varepsilon \in (0, 1)$ , the subcollection,  $\mathcal{A}$ , of the  $\lceil kr(\phi + 1)/\varepsilon \rceil$ -largest sets of  $\mathcal{V}$  (by size) contains a subcollection of at most  $k$  sets with unique coverage at least  $(1 - \varepsilon)\text{OPT}$ .*

**Proof.** Assume that  $|\mathcal{V}| \geq \lceil kr(\phi + 1)/\varepsilon \rceil$ : otherwise,  $\mathcal{A}$  would contain every set in  $\mathcal{V}$  and so would trivially have  $\mathcal{O}$  as a subcollection. Let  $\mathcal{O}^{\text{in}} = \mathcal{O} \cap \mathcal{A}$  and  $\mathcal{O}^{\text{out}} = \mathcal{O} \setminus \mathcal{A}$ . Let  $\mathcal{Z}$  be a uniform random sample of  $|\mathcal{O}^{\text{out}}|$  sets chosen from  $\mathcal{A}$  without replacement. The main goal is to prove Claim 3.5, below. Since  $\mathcal{O}^{\text{in}}$  and  $\mathcal{Z}$  are subsets of  $\mathcal{A}$ , this implies the existence of subcollection  $\mathcal{B} \subseteq \mathcal{A}$  as required by the lemma.

We start with the following lower bound on the expected unique coverage of  $\mathcal{O}^{\text{in}} \cup \mathcal{Z}$ , as shown in inequality (1) below. Then we lower bound each of the RHS terms separately and simplify afterwards. By definition,

$$|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})| \geq |\tilde{\psi}(\mathcal{O}^{\text{in}})| + |\tilde{\psi}(\mathcal{Z})| - (|\tilde{\psi}(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})| + |\psi(\mathcal{O}^{\text{in}}) \cap \tilde{\psi}(\mathcal{Z})|),$$

hence, by linearity of expectation,

$$\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|] \geq |\tilde{\psi}(\mathcal{O}^{\text{in}})| + \mathbb{E}[|\tilde{\psi}(\mathcal{Z})|] - \mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|] - \mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \tilde{\psi}(\mathcal{Z})|]. \quad (1)$$

Define an intermediate error parameter,  $\varepsilon' = \varepsilon/(\phi + 1)$ , meaning  $|\mathcal{A}| = \lceil kr/\varepsilon' \rceil$ . The probability of a set  $S \in \mathcal{A}$  being selected in  $\mathcal{Z}$  is  $p := |\mathcal{O}^{\text{out}}|/|\mathcal{A}| \leq k/(kr/\varepsilon') = \varepsilon'/r$ . Now Claim 3.2, below, is easily derived from the proof of Lemma 11 in by McGregor et al. [15].

► **Claim 3.2.** It holds that  $\mathbb{E}[|\tilde{\psi}(\mathcal{Z})|] \geq (1 - \varepsilon')|\tilde{\psi}(\mathcal{O}^{\text{out}})|$ .

**Proof.** Quantity  $|\tilde{\psi}(\mathcal{Z})|$  can be lower bounded by summing, over every  $S \in \mathcal{Z}$ , the number of elements in  $S$  not contained in any other  $T \in \mathcal{Z} \setminus \{S\}$ . From there, we prove inequality (3.2), below. We let  $[\mathcal{E}]$  denote the indicator variable for event  $\mathcal{E}$ .

$$\begin{aligned}
 |\tilde{\psi}(\mathcal{Z})| &\geq \sum_{S \in \mathcal{Z}} \left( |S| - \sum_{T \in \mathcal{Z} \setminus \{S\}} |S \cap T| \right), \text{ hence,} \\
 \mathbb{E}[|\tilde{\psi}(\mathcal{Z})|] &\geq \mathbb{E} \left[ \sum_{S \in \mathcal{A}} \left( |S| \mathbb{1}[S \in \mathcal{Z}] - \sum_{T \in \mathcal{A} \setminus \{S\}} |S \cap T| \mathbb{1}[S \in \mathcal{Z} \wedge T \in \mathcal{Z}] \right) \right] \\
 &\geq \sum_{S \in \mathcal{A}} \left( |S|p - \sum_{T \in \mathcal{A} \setminus \{S\}} |S \cap T|p^2 \right) && \Pr[S \in \mathcal{Z} \wedge T \in \mathcal{Z}] \leq p^2 \\
 &\geq \sum_{S \in \mathcal{A}} (|S|p - |S|p^2(r-1)) \geq p(1-pr) \sum_{S \in \mathcal{A}} |S| && \begin{array}{l} \text{each } x \in S \text{ intersects} \\ \leq r-1 \text{ other sets} \end{array} \\
 &\geq p(1-\varepsilon') \sum_{S \in \mathcal{A}} |S| && p \leq \frac{\varepsilon'}{r} \\
 &\geq p(1-\varepsilon') |\mathcal{A}| \frac{\sum_{Y \in \mathcal{O}^{\text{out}}} |Y|}{|\mathcal{O}^{\text{out}}|} && \begin{array}{l} \text{for all } S \in \mathcal{A} \text{ and all} \\ Y \in \mathcal{O}^{\text{out}}: |S| \geq |Y| \end{array} \\
 &\geq p(1-\varepsilon') |\mathcal{A}| \frac{|\tilde{\psi}(\mathcal{O}^{\text{out}})|}{|\mathcal{O}^{\text{out}}|} && \text{subadditivity of } \tilde{\psi} \\
 &= p(1-\varepsilon') \frac{|\tilde{\psi}(\mathcal{O}^{\text{out}})|}{p} = (1-\varepsilon') |\tilde{\psi}(\mathcal{O}^{\text{out}})|. && \triangleleft
 \end{aligned}$$

Claim 3.3 upper bounds the expected size of the overlap between  $\tilde{\psi}(\mathcal{O}^{\text{in}})$  and  $\psi(\mathcal{Z})$  and the expected size of the overlap between  $\tilde{\psi}(\mathcal{Z})$  and  $\psi(\mathcal{O}^{\text{in}})$ .

▷ **Claim 3.3.**  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|] \leq \varepsilon' |\tilde{\psi}(\mathcal{O}^{\text{in}})|$  and  $\mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \tilde{\psi}(\mathcal{Z})|] \leq \varepsilon' |\psi(\mathcal{O}^{\text{in}})|$ .

*Proof.* To prove the first inequality,

$$\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|] \leq \sum_{x \in \tilde{\psi}(\mathcal{O}^{\text{in}})} \sum_{S \in \mathcal{A}: x \in S} \Pr[S \in \mathcal{Z}] \leq \sum_{x \in \tilde{\psi}(\mathcal{O}^{\text{in}})} rp \leq \varepsilon' |\tilde{\psi}(\mathcal{O}^{\text{in}})|.$$

To prove the second inequality, it is clear that  $\tilde{\psi}(\mathcal{Z}) \subseteq \psi(\mathcal{Z})$  for all  $\mathcal{Z}$ , so we have  $\mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \tilde{\psi}(\mathcal{Z})|] \leq \mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|]$ . Then substituting  $\psi(\mathcal{O}^{\text{in}})$  for  $\tilde{\psi}(\mathcal{O}^{\text{in}})$  in the argument for the first inequality, we see that  $\mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|] \leq \varepsilon' |\psi(\mathcal{O}^{\text{in}})|$ .  $\triangleleft$

We now turn to a property of the optimal solution for **Max Unique Coverage**,  $\mathcal{O}$ .

▷ **Claim 3.4.**  $\phi |\tilde{\psi}(\mathcal{O})| \geq |\psi(\mathcal{O})|$ .

*Proof.* Recall that we assumed that every collection of sets has unique coverage ratio at most  $\phi$ . In particular,  $\mathcal{O}$  has a subcollection,  $\mathcal{Q}$ , of at most  $k$  sets with  $\phi |\tilde{\psi}(\mathcal{Q})| \geq |\psi(\mathcal{O})|$ . By optimality,  $\mathcal{O}$ 's unique coverage is at least that of  $\mathcal{Q}$ . Thus, we get the desired inequality.  $\triangleleft$

Starting from Ineq. (1), we can now lower bound  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|]$ , thus proving the lemma.

▷ **Claim 3.5.** We have the lower bound  $\mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|] \geq (1-\varepsilon) |\tilde{\psi}(\mathcal{O})|$ .

Proof.

$$\begin{aligned}
& \mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}} \cup \mathcal{Z})|] \\
& \geq |\tilde{\psi}(\mathcal{O}^{\text{in}})| + \mathbb{E}[|\tilde{\psi}(\mathcal{Z})|] - \mathbb{E}[|\tilde{\psi}(\mathcal{O}^{\text{in}}) \cap \psi(\mathcal{Z})|] - \mathbb{E}[|\psi(\mathcal{O}^{\text{in}}) \cap \tilde{\psi}(\mathcal{Z})|] && \text{Ineq. (1)} \\
& \geq |\tilde{\psi}(\mathcal{O}^{\text{in}})| + (1 - \varepsilon')|\tilde{\psi}(\mathcal{O}^{\text{out}})| - \varepsilon'|\tilde{\psi}(\mathcal{O}^{\text{in}})| - \varepsilon'|\psi(\mathcal{O}^{\text{in}})| && \text{Claims 3.2 and 3.3} \\
& = (1 - \varepsilon')(|\tilde{\psi}(\mathcal{O}^{\text{in}})| + |\tilde{\psi}(\mathcal{O}^{\text{out}})|) - \varepsilon'|\psi(\mathcal{O}^{\text{in}})| \\
& \geq (1 - \varepsilon')|\tilde{\psi}(\mathcal{O})| - \varepsilon'|\psi(\mathcal{O}^{\text{in}})| && \text{subadditivity of } \tilde{\psi} \\
& \geq (1 - \varepsilon')|\tilde{\psi}(\mathcal{O})| - \varepsilon'|\psi(\mathcal{O})| && \text{monotonicity of } \psi \\
& \geq (1 - \varepsilon')|\tilde{\psi}(\mathcal{O})| - \varepsilon'\phi|\tilde{\psi}(\mathcal{O})| && \text{Claim 3.4} \\
& = (1 - \varepsilon'(1 + \phi))|\tilde{\psi}(\mathcal{O})| \\
& = (1 - \varepsilon)|\tilde{\psi}(\mathcal{O})|. && \triangleleft
\end{aligned}$$

### 3.2 Applications of the Kernelization Lemma

We now apply the Kernelization Lemma to prove the following theorem.

► **Theorem 3.6.** *Suppose that every collection of sets has unique coverage ratio at most  $\phi$ . Let  $\mathcal{V}$  denote a collection of sets,  $k \geq 2$  denote the cardinality constraint,  $r \geq 2$  denote the maximum frequency in  $\mathcal{V}$ , and  $\varepsilon \in (0, 1)$  denote an error parameter. Then, there exists*

1. *an FPT-AS that finds a  $(1 - \varepsilon)$ -approximation for **Max Unique Coverage** and has a running time of  $(er(\phi + 1)/\varepsilon)^k \text{poly}(m, n, 1/\varepsilon)$ ; and*
2. *a streaming algorithm that finds a  $(1 - 3\varepsilon)$ -approximation for **Max Unique Coverage** with probability  $1 - 1/\text{poly}(m)$  and uses  $\tilde{O}(\phi k^2 r/\varepsilon^3)$  space.*

Our algorithm, **UNIQUE TOPSETS**, takes a collection of sets  $\mathcal{V}$  with maximum frequency  $r \geq 2$ , a cardinality constraint  $k$ , and an error parameter  $\varepsilon \in (0, 1)$ , and returns a  $(1 - \varepsilon)$ -approximation for **Max Unique Coverage**. It also takes parameter  $\phi$ , an upper bound on the unique coverage ratio of every subcollection. **UNIQUE TOPSETS** first finds  $\mathcal{A}$ , the  $\lceil kr(\phi + 1)/\varepsilon \rceil$ -largest sets  $S \in \mathcal{V}$  by size  $|S|$ . Then, it brute-forces over  $\mathcal{A}$ , i.e., it finds the subcollection of  $\mathcal{A}$  containing at most  $k$  sets and has the maximum unique coverage.

**FPT-AS.** Let us first see how **UNIQUE TOPSETS** has the properties of the FPT-AS claimed in Theorem 3.6. The Kernelization Lemma (Lemma 3.1) immediately implies that the solution returned by **UNIQUE TOPSETS** is a  $(1 - \varepsilon)$ -approximation. The running time bound follows by bounding the number of subcollections of  $\mathcal{A}$  containing at most  $k$  sets.

► **Lemma 3.7.** ***UNIQUE TOPSETS** has running time in  $(er(\phi + 1)/\varepsilon)^k \text{poly}(m, n, 1/\varepsilon)$ .*

**Proof.** **UNIQUE TOPSETS** considers every possible collection of  $\ell \in [k]$  sets from  $\mathcal{A}$  and outputs the one with the best unique coverage. Below, the second inequality holds since replacing  $\ell$  with  $k$  makes each binomial coefficient larger, as  $\ell \leq k \leq kr/2$  due to  $r \geq 2$ ; the equality holds since  $\binom{z+1}{k} = \binom{z}{k}(z+1)/(z+1-k)$ ; and the final inequality holds since  $\binom{z}{k} \leq (ez/k)^k$ . Thus, the running time is bounded as follows.

$$\begin{aligned}
& \sum_{\ell=1}^k \binom{|\mathcal{A}|}{\ell} \text{poly}(m, n) \leq \text{poly}(m, n) \sum_{\ell=1}^k \left( \frac{kr(\phi+1)}{\varepsilon} \frac{1}{\ell} + 1 \right) \leq \text{poly}(m, n) k \left( \frac{kr(\phi+1)}{\varepsilon} \frac{1}{k} + 1 \right) \\
& = \text{poly}(m, n, 1/\varepsilon) \left( \frac{kr(\phi+1)}{\varepsilon} \right) \leq \text{poly}(m, n, 1/\varepsilon) \left( \frac{er(\phi+1)}{\varepsilon} \right)^k. && \triangleleft
\end{aligned}$$

**Streaming Algorithm.** UNIQUETOPSETS can also be run on a data stream using the subsampling approach from Lemma 2.1. UNIQUETOPSETS returns a  $(1 - \varepsilon)$ -approximation in each subsampled instance by Lemma 3.1, implying a  $(1 - 3\varepsilon)$ -approximation for the original instance,  $\mathcal{V}$ , with probability  $1 - 1/\text{poly}(m)$ . Further, it stores  $|\mathcal{A}| \leq \lceil kr(\phi + 1)/\varepsilon \rceil$  sets in each subsampled instance, implying an overall space complexity of  $\lceil \log_2 n \rceil \cdot |\mathcal{A}| \cdot \tilde{O}(k/\varepsilon^2) = \tilde{O}(\phi k^2 r / \varepsilon^3)$ .

### 3.3 Polynomial-Time Streaming Algorithm

Here we present a single-pass streaming algorithm that returns a  $(1/(2\phi) - \varepsilon)$ -approximation for **Max Unique Coverage**, given a bound on the unique coverage ratio,  $\phi$ . We present the algorithm in Theorem 3.8 below.

In the theorem statement, we assume we can use an offline polynomial-time algorithm, ALG, that takes a collection  $\mathcal{C}$  and returns a subcollection  $\mathcal{B} \subseteq \mathcal{C}$  such that  $|\psi(\mathcal{B})| \geq |\psi(\mathcal{C})|/\phi$  for a ratio  $\phi$  depending on  $|\mathcal{C}| \leq k$ , the maximum frequency  $r$ , and the maximum set size  $d$ . ALG can be substituted with a procedure that runs all of our unique coverage algorithms from Section 4 on  $\mathcal{C}$  and returns the solution with the best unique coverage.

► **Theorem 3.8.** *Let  $\mathcal{V}$  denote a data stream of  $m$  sets,  $k \geq 2$  denote a cardinality constraint,  $r \geq 2$  denote the maximum frequency in  $\mathcal{V}$ ,  $d \geq 2$  denote the maximum set size in  $\mathcal{V}$ , and  $\varepsilon \in (0, 1)$  denote an error parameter. Further, assume we have a polynomial-time algorithm ALG with unique coverage ratio  $\phi$  depending on  $k, r$ , and  $d$ . Then we can find a  $(1/(2\phi) - 3\varepsilon)$ -approximation for **Max Unique Coverage** with probability  $1 - 1/\text{poly}(m)$ , using one pass,  $\tilde{O}(k^2/\varepsilon^3)$  space, and in polynomial-time.*

**Proof.** We use the subsampling approach from Lemma 2.1. In each subsampled instance, we use an existing polynomial-time streaming algorithm [16] to find a  $(1/2 - \varepsilon)$ -approximation,  $\mathcal{C}$ , for **Max Coverage** in one pass while storing  $\tilde{O}(k/\varepsilon)$  sets; the sets in  $\mathcal{C}$  must be stored explicitly so that we can run ALG on  $\mathcal{C}$ . Running ALG on  $\mathcal{C}$  returns a  $\mathcal{B} \subseteq \mathcal{C}$  that is a  $(1/(2\phi) - \varepsilon)$ -approximation for the subsampled instance of **Max Unique Coverage**. This implies a  $(1/(2\phi) - 3\varepsilon)$ -approximation for the original instance,  $\mathcal{V}$ , with probability  $1 - 1/\text{poly}(m)$ . Further, explicitly storing  $\tilde{O}(k/\varepsilon)$  sets in each subsampled instance implies an overall space complexity of  $\lceil \log_2 n \rceil \cdot \tilde{O}(k/\varepsilon) \cdot \tilde{O}(k/\varepsilon^2) = \tilde{O}(k^2/\varepsilon^3)$ . ◀

## 4 Algorithms for Bounding the Unique Coverage Ratio

We here present algorithms that run in polynomial time. Given a collection,  $\mathcal{C}$ , each returns a subcollection  $\mathcal{B} \subseteq \mathcal{C}$  such that  $\mathcal{B}$ 's unique coverage is within a logarithmic ratio of  $\mathcal{C}$ 's coverage. We hence call this the *unique coverage ratio* of an algorithm. Our algorithms UNIQUEGREEDY (Section 4.1), UNIQUEGREEDYFREQ (Section 4.2), and UNIQUEGREEDYSIZE (Section 4.3) have unique coverage ratios that are logarithmic in  $\ell = |\mathcal{C}|$ ,  $r$ , and  $d$ , respectively.

### 4.1 UniqueGreedy

We present and analyze our algorithm UNIQUEGREEDY, with pseudocode in Algorithm 1. Its purpose is to take a collection  $\mathcal{C}$  of  $\ell$  sets and return a collection  $\mathcal{B} \subseteq \mathcal{C}$  whose *unique coverage* is at least a  $1/H_\ell$  factor of  $\mathcal{C}$ 's coverage. We formally state this in Theorem 4.1.

**UniqueGreedy Overview.** UNIQUEGREEDY first checks whether  $\mathcal{C}$ 's unique coverage is at least  $1/H_\ell$  of its own coverage. If so, then it immediately returns  $\mathcal{C}$  as the solution, which of course occurs if  $\ell = 1$ . If not, then the idea is to discard the set  $T \in \mathcal{C}$  with the smallest

contribution to  $\mathcal{C}$ 's unique coverage. It follows that the total loss in coverage from  $\mathcal{C}$  to  $\mathcal{C} \setminus \{T\}$  is only  $1/\ell$  of  $\mathcal{C}$ 's unique coverage. Observe that  $T$  contributes at most  $1/\ell$  to  $\mathcal{C}$ 's unique coverage, and any elements in  $T$  that are also in  $\mathcal{C}$ 's non-unique cover must remain in  $\mathcal{C} \setminus \{T\}$ 's cover. We then apply `UNIQUEGREEDY` recursively, to  $\mathcal{C} \setminus \{T\}$ . As we show in Theorem 4.1, since the performance of `UNIQUEGREEDY` relates unique coverage to coverage,  $\mathcal{C} \setminus \{T\}$  has sufficient coverage so that the recursive solution from `UNIQUEGREEDY`( $\mathcal{C} \setminus \{T\}$ ) has a unique coverage of at least  $1/H_\ell$  of  $\mathcal{C}$ 's coverage.

■ **Algorithm 1** `UNIQUEGREEDY`.

---

**Input:**  $\mathcal{C}$ : collection of  $\ell$  sets.

**Output:**  $\mathcal{B} \subseteq \mathcal{C}$ : subcollection satisfying  $|\tilde{\psi}(\mathcal{B})| \geq |\psi(\mathcal{C})|/H_\ell$ .

```

1 if  $|\tilde{\psi}(\mathcal{C})| \geq |\psi(\mathcal{C})|/H_\ell$  then
2    $\mathcal{B} \leftarrow \mathcal{C}$ 
3 else
4    $T \leftarrow \arg \min_{S \in \mathcal{C}} |S \cap \tilde{\psi}(\mathcal{C})|$ 
5    $\mathcal{B} \leftarrow \text{UNIQUEGREEDY}(\mathcal{C} \setminus \{T\})$ 
6 return  $\mathcal{B}$ 

```

---

► **Theorem 4.1.** *Given a collection of  $\ell$  sets,  $\mathcal{C}$ , `UNIQUEGREEDY` returns a collection  $\mathcal{B} \subseteq \mathcal{C}$  satisfying*

$$|\tilde{\psi}(\mathcal{B})| \geq \frac{|\psi(\mathcal{C})|}{H_\ell}. \quad (2)$$

**Proof.** We prove Theorem 4.1 by induction on  $\ell = |\mathcal{C}|$ .

**Base Case.** If  $\ell = 1$ , then  $|\tilde{\psi}(\mathcal{C})| = |\psi(\mathcal{C})|$  and  $\mathcal{B} = \mathcal{C}$ , so we are done.

**Inductive Case.** Consider the case  $\ell \geq 2$ , and assume that Theorem 4.1 holds for  $\ell - 1$ . Then one of two subcases must hold: (i)  $|\tilde{\psi}(\mathcal{C})| \geq |\psi(\mathcal{C})|/H_\ell$ ; or (ii) the negation,  $|\tilde{\psi}(\mathcal{C})| < |\psi(\mathcal{C})|/H_\ell$ . In subcase (i), the Line 1 condition succeeds and `UNIQUEGREEDY` returns the subcollection  $\mathcal{B} = \mathcal{C}$ , which clearly satisfies Ineq. (2).

So, we focus on subcase (ii); since  $|\tilde{\psi}(\mathcal{C})| < |\psi(\mathcal{C})|/H_\ell$ , the Line 1 condition fails, thus Line 5 assigns to  $\mathcal{B}$  the solution from the recursive call on  $\mathcal{C} \setminus \{T\}$ . Claim 4.3 lower bounds the coverage of this subcollection,  $|\psi(\mathcal{C} \setminus \{T\})|$ . Prior to that, we prove a handy claim.

► **Claim 4.2.**  $|\psi(\mathcal{C} \setminus \{T\})| = |\psi_{\geq 2}(\mathcal{C})| + |\tilde{\psi}(\mathcal{C}) \setminus T|$ .

**Proof.** Observe that  $\psi_{\geq 2}(\mathcal{C})$  and  $\tilde{\psi}(\mathcal{C}) \setminus T$  are disjoint; so it suffices to show that  $\psi(\mathcal{C} \setminus \{T\}) = \psi_{\geq 2}(\mathcal{C}) \cup (\tilde{\psi}(\mathcal{C}) \setminus T)$ . We first show that RHS is a subset of LHS. Each element covered at least twice in  $\mathcal{C}$  remains covered in  $\mathcal{C} \setminus \{T\}$ ; while each element uniquely covered in  $\mathcal{C}$  that is not in  $T$  remains covered in  $\mathcal{C} \setminus \{T\}$ . Going the other way, consider an element that is in neither  $\psi_{\geq 2}(\mathcal{C})$  nor  $\tilde{\psi}(\mathcal{C}) \setminus T$ : then the only set it was in was  $T$ , and hence it is not in  $\mathcal{C} \setminus \{T\}$ .  $\triangleleft$

► **Claim 4.3.** Subcollection  $\mathcal{C} \setminus \{T\}$  satisfies

$$|\psi(\mathcal{C} \setminus \{T\})| \geq \left(1 - \frac{1}{\ell H_\ell}\right) |\psi(\mathcal{C})|.$$

## 25:12 Maximum Unique Coverage on Streams

Proof. First, observe that the contribution of each  $S \in \mathcal{C}$  to  $\tilde{\psi}(\mathcal{C})$ , i.e.,  $|S \cap \tilde{\psi}(\mathcal{C})|$ , is disjoint from the contributions of all other sets in  $\mathcal{C}$ : each element in  $\tilde{\psi}(\mathcal{C})$  is covered by exactly one set. Therefore, the set  $T = \arg \min_{S \in \mathcal{C}} |S \cap \tilde{\psi}(\mathcal{C})|$  in Line 4 satisfies

$$|T \cap \tilde{\psi}(\mathcal{C})| \leq \frac{|\tilde{\psi}(\mathcal{C})|}{\ell}. \quad (3)$$

With Claim 4.2, we now prove Claim 4.3.

$$\begin{aligned} |\psi(\mathcal{C} \setminus \{T\})| &= |\psi_{\geq 2}(\mathcal{C})| + |\tilde{\psi}(\mathcal{C}) \setminus T| \\ &= |\psi(\mathcal{C})| - |\tilde{\psi}(\mathcal{C})| + |\tilde{\psi}(\mathcal{C})| - |T \cap \tilde{\psi}(\mathcal{C})| \\ &= |\psi(\mathcal{C})| - |T \cap \tilde{\psi}(\mathcal{C})| \\ &\geq |\psi(\mathcal{C})| - \frac{|\tilde{\psi}(\mathcal{C})|}{\ell} && \text{Ineq. (3)} \\ &> |\psi(\mathcal{C})| - \frac{|\psi(\mathcal{C})|}{\ell H_\ell} && \text{subcase (ii)} \\ &= \left(1 - \frac{1}{\ell H_\ell}\right) |\psi(\mathcal{C})|. && \triangleleft \end{aligned}$$

Recall that in subcase (ii), Line 5 assigns to  $\mathcal{B}$  the solution from the recursive call on  $\mathcal{C} \setminus \{T\}$ . Since  $|\mathcal{C} \setminus \{T\}| = \ell - 1$ , we apply the inductive assumption to prove that  $\mathcal{B}$  satisfies Ineq. (2).

$$\begin{aligned} |\tilde{\psi}(\mathcal{B})| &\geq \frac{|\psi(\mathcal{C} \setminus \{T\})|}{H_{\ell-1}} && \text{inductive assumption} \\ &\geq \frac{1}{H_{\ell-1}} \left(1 - \frac{1}{\ell H_\ell}\right) |\psi(\mathcal{C})| && \text{Claim 4.3} \\ &= \frac{1}{H_{\ell-1}} \frac{\ell H_\ell - 1}{\ell H_\ell} |\psi(\mathcal{C})| \\ &= \frac{1}{H_{\ell-1}} \frac{H_\ell - \frac{1}{\ell}}{H_\ell} |\psi(\mathcal{C})| \\ &= \frac{|\psi(\mathcal{C})|}{H_\ell}. && H_\ell - \frac{1}{\ell} = H_{\ell-1}, \text{ for } \ell \geq 2 \end{aligned}$$

We have proven that  $\mathcal{B}$  satisfies Ineq. (2) in the base case and the inductive case, proving Theorem 4.1.  $\blacktriangleleft$

## 4.2 UniqueGreedyFreq

In this section, we present and analyze our algorithm UNIQUEGREEDYFREQ, with pseudocode in Algorithm 2. The purpose of this algorithm is to take a collection  $\mathcal{C}$  with maximum frequency  $r \leq |\mathcal{C}|$ , and an error parameter  $\varepsilon_r \in (0, 1)$ , and return a collection  $\mathcal{B} \subseteq \mathcal{C}$  whose unique coverage is at least a  $(1/H_{\lceil r(r-1)/\varepsilon_r \rceil} - \varepsilon_r)$  factor of  $\mathcal{C}$ 's coverage. By an appropriate choice of  $\varepsilon_r$  depending on  $r$ , this factor can be simplified to  $1/(2 \ln r + o(\log r))$ .

**UniqueGreedyFreq Overview.** The idea of UNIQUEGREEDYFREQ is to group all of the sets from  $\mathcal{C}$  into  $\hat{\ell}$  disjoint collections,  $\mathcal{G}_1, \dots, \mathcal{G}_{\hat{\ell}}$ , so that the sets must be selected into the solution  $\mathcal{B}$  in these groups, i.e., for each  $i \in [\hat{\ell}]$ , either all of the sets in  $\mathcal{G}_i$ , or none of the sets in  $\mathcal{G}_i$ , must be selected into  $\mathcal{B}$ . Then, letting  $\hat{\mathcal{C}}$  be the collection of the covers of  $\mathcal{G}_1, \dots, \mathcal{G}_{\hat{\ell}}$ , we can call UNIQUEGREEDY on  $\hat{\mathcal{C}}$  to find a selection of these covers, namely  $\hat{\mathcal{B}}$ . The returned solution,  $\mathcal{B}$ , is constructed by merging each  $\mathcal{G}_i$  whose cover was selected into  $\hat{\mathcal{B}}$ , which ensures that the sets are selected in groups.



It can be seen that, by calling UNIQUEGREEDY on  $\hat{\mathcal{C}}$  and by Theorem 4.1, the unique coverage of  $\hat{\mathcal{B}}$  is at least  $1/H_{\hat{\ell}}$  of  $\hat{\mathcal{C}}$ 's coverage, and therefore at least  $1/H_{\hat{\ell}}$  of  $\mathcal{C}$ 's coverage since  $\hat{\mathcal{C}}$  and  $\mathcal{C}$  have the same cover. The issue now is that sets from the same  $\mathcal{G}_i$  can overlap after being selected as a group into  $\mathcal{B}$ , which would make  $\mathcal{B}$ 's unique coverage smaller than  $\hat{\mathcal{B}}$ 's unique coverage. This is addressed by setting the number of groups to be  $\hat{\ell} = \lceil r(r-1)/\varepsilon_r \rceil$ , and by the way UNIQUEGREEDYFREQ allocates the sets into these groups: it allocates each  $S \in \mathcal{C}$  to the group  $\mathcal{G}_i$  whose unique coverage intersects the least with  $S$ . In this way, the total unique coverage that is lost due to overlapping sets in the same  $\mathcal{G}_i$  can be bounded by  $\varepsilon_r |\psi(\mathcal{C})|$ . Thus,  $\mathcal{B}$ 's unique coverage is at least  $(1/H_{\hat{\ell}} - \varepsilon_r) = (1/H_{\lceil r(r-1)/\varepsilon_r \rceil} - \varepsilon_r)$  of  $\mathcal{C}$ 's coverage. Details are given in the proof of Theorem 4.4.

■ **Algorithm 2** UNIQUEGREEDYFREQ.

---

**Input:**  $\mathcal{C}$ : collection with maximum frequency  $r \geq 2$ ,  $\varepsilon_r \in (0, 1)$ : error parameter.  
**Output:**  $\mathcal{B} \subseteq \mathcal{C}$ : collection satisfying  $|\tilde{\psi}(\mathcal{B})| \geq (1/H_{\lceil r(r-1)/\varepsilon_r \rceil} - \varepsilon_r) |\psi(\mathcal{C})|$ .

```

1  $\hat{\ell} \leftarrow \lceil r(r-1)/\varepsilon_r \rceil$ 
2 for  $i \in [\hat{\ell}]$  do // Initialize empty groups
3    $\mathcal{G}_i \leftarrow \emptyset$ 
4 for  $S \in \mathcal{C}$  do // Allocate sets to groups
5    $i \leftarrow \arg \min_{j \in [\hat{\ell}]} |\tilde{\psi}(\mathcal{G}_j) \cap S|$ 
6    $\mathcal{G}_i \leftarrow \mathcal{G}_i \cup \{S\}$ 
7  $\hat{\mathcal{C}} \leftarrow \{\psi(\mathcal{G}_1), \dots, \psi(\mathcal{G}_{\hat{\ell}})\}$  // Define collection of groups' covers
8  $\hat{\mathcal{B}} \leftarrow \text{UNIQUEGREEDY}(\hat{\mathcal{C}})$ 
9  $\mathcal{B} \leftarrow \emptyset$ 
10 for  $\psi(\mathcal{G}_i) \in \hat{\mathcal{B}}$  do // Construct returned solution
11    $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{G}_i$ 
12 return  $\mathcal{B}$ 

```

---

► **Theorem 4.4.** *Given  $\mathcal{C}$  with maximum frequency  $r \geq 2$ , and error parameter  $\varepsilon_r \in (0, 1)$ , algorithm UNIQUEGREEDYFREQ returns a collection  $\mathcal{B} \subseteq \mathcal{C}$  satisfying*

$$|\tilde{\psi}(\mathcal{B})| \geq \left( \frac{1}{H_{\lceil r(r-1)/\varepsilon_r \rceil}} - \varepsilon_r \right) |\psi(\mathcal{C})|. \quad (4)$$

Moreover, setting  $\varepsilon_r = (9.27 \ln r)^{-1} (2 \ln r + 2 \ln \ln r + 5.61)^{-1}$ , we obtain

$$|\tilde{\psi}(\mathcal{B})| \geq \left( \frac{1 - 1/(9.27 \ln r)}{2 \ln r + 2 \ln \ln r + 5.61} \right) |\psi(\mathcal{C})| \geq \frac{1}{2 \ln r + o(\log r)} |\psi(\mathcal{C})|. \quad (5)$$

**Proof.** We first prove Ineq. (4), starting with the following claim.

► **Claim 4.5.**  $\tilde{\psi}(\mathcal{B}) = \tilde{\psi}(\hat{\mathcal{B}}) \setminus \bigcup_{i: \psi(\mathcal{G}_i) \in \hat{\mathcal{B}}} \psi_{\geq 2}(\mathcal{G}_i)$ .

**Proof.** Consider an element  $x$  that is in exactly one set in  $\mathcal{B}$ . This means that  $x$  is in exactly one set from exactly one group, say  $\mathcal{G}_y$ , chosen in  $\mathcal{B}$ . Focusing on  $\hat{\mathcal{B}}$ , element  $x$  is clearly in  $\psi(\mathcal{G}_y)$  only, but might occur more than once in  $\mathcal{G}_y$ . Excluding elements that are in  $\psi_{\geq 2}(\mathcal{G}_i)$  for every  $i$ , we thus have the claim statement. ◁

With Claim 4.5, we prove Claim 4.6.

► **Claim 4.6.** The solution  $\mathcal{B}$  satisfies  $|\tilde{\psi}(\mathcal{B})| \geq |\tilde{\psi}(\hat{\mathcal{B}})| - \varepsilon_r |\psi(\mathcal{C})|$ .

## 25:14 Maximum Unique Coverage on Streams

Proof. Given Claim 4.5,  $\mathcal{B}$  satisfies Ineq. (6),

$$\begin{aligned} |\tilde{\psi}(\mathcal{B})| &\geq |\tilde{\psi}(\hat{\mathcal{B}})| - \sum_{i: \psi(\mathcal{G}_i) \in \hat{\mathcal{B}}} |\psi_{\geq 2}(\mathcal{G}_i)| \\ &\geq |\tilde{\psi}(\hat{\mathcal{B}})| - \sum_{i \in [\hat{\ell}]} |\psi_{\geq 2}(\mathcal{G}_i)|. \end{aligned} \quad (6)$$

We upper bound  $\sum_{i \in [\hat{\ell}]} |\psi_{\geq 2}(\mathcal{G}_i)|$  in Ineq. (6). Let  $S_t$  be the  $t^{\text{th}}$  set allocated in the Line 4 loop, let  $\mathcal{G}_{i,0} = \emptyset$ , and let  $\mathcal{G}_{i,t}$  be the subcollection  $\mathcal{G}_i$  just after allocating  $S_t$ .

Upon inserting  $S_t$  into  $\mathcal{G}_i$ , every element in  $\tilde{\psi}(\mathcal{G}_{i,t-1})$  that becomes non-uniquely covered is accounted for by  $\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t$ . So it holds that  $|\psi_{\geq 2}(\mathcal{G}_{i,t})| - |\psi_{\geq 2}(\mathcal{G}_{i,t-1})| = |\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t|$ . Thus,  $|\psi_{\geq 2}(\mathcal{G}_i)|$  can be expressed by Equation (7) below, observing that for  $S_t$  the relevant difference term is zero.

$$\begin{aligned} |\psi_{\geq 2}(\mathcal{G}_i)| &= \sum_{S_t \in \mathcal{G}_i} (|\psi_{\geq 2}(\mathcal{G}_{i,t})| - |\psi_{\geq 2}(\mathcal{G}_{i,t-1})|) && \text{telescoping series} \\ &= \sum_{S_t \in \mathcal{G}_i} |\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t|. \end{aligned} \quad (7)$$

For each  $i \in [\hat{\ell}]$  and each  $S_t \in \mathcal{G}_i$ , we want to show an upper bound of  $|\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t| \leq (r-1)|S_t|/\hat{\ell}$ . To see this, since the maximum frequency is  $r$ , each element  $x \in S_t$  is covered by at most  $r-1$  other sets, each possibly in a different group. Therefore, we have that

$$\begin{aligned} \sum_{j \in [\hat{\ell}]} |\tilde{\psi}(\mathcal{G}_{j,t-1}) \cap \{x\}| &\leq r-1, \\ \sum_{x \in S_t} \sum_{j \in [\hat{\ell}]} |\tilde{\psi}(\mathcal{G}_{j,t-1}) \cap \{x\}| &\leq \sum_{x \in S_t} (r-1), \\ \sum_{j \in [\hat{\ell}]} |\tilde{\psi}(\mathcal{G}_{j,t-1}) \cap S_t| &\leq (r-1)|S_t|. \end{aligned}$$

Recall that  $S_t$  was allocated to the group  $\mathcal{G}_i = \arg \min_{j \in [\hat{\ell}]} |\tilde{\psi}(\mathcal{G}_{j,t-1}) \cap S_t|$  in Lines 5–6. Therefore, by averaging on the above inequality, we have that for each  $i \in [\hat{\ell}]$  and each  $S_t$  that ends up in  $\mathcal{G}_i$ ,

$$|\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t| \leq \frac{r-1}{\hat{\ell}} |S_t|. \quad (8)$$

Now we upper bound  $\sum_{i \in [\hat{\ell}]} |\psi_{\geq 2}(\mathcal{G}_i)|$ .

$$\begin{aligned} \sum_{i \in [\hat{\ell}]} |\psi_{\geq 2}(\mathcal{G}_i)| &= \sum_{i \in [\hat{\ell}]} \sum_{S_t \in \mathcal{G}_i} |\tilde{\psi}(\mathcal{G}_{i,t-1}) \cap S_t| && \text{Equation (7)} \\ &\leq \frac{r-1}{\hat{\ell}} \sum_{i \in [\hat{\ell}]} \sum_{S_t \in \mathcal{G}_i} |S_t| && \text{Ineq. (8)} \\ &= \frac{r-1}{\hat{\ell}} \sum_{S \in \mathcal{C}} |S| && \mathcal{G}_1, \dots, \mathcal{G}_{\hat{\ell}} \text{ partitions } \mathcal{C} \\ &\leq \frac{r-1}{\hat{\ell}} r |\psi(\mathcal{C})| && \text{for all } x \in \psi(\mathcal{C}): \text{freq}_{\mathcal{C}}(x) \leq r \\ &= \frac{r(r-1)}{\lceil r(r-1)/\varepsilon_r \rceil} |\psi(\mathcal{C})| && \text{value of } \hat{\ell} \text{ (Line 1)} \end{aligned}$$

$$\begin{aligned} &\leq \frac{r(r-1)}{r(r-1)/\varepsilon_r} |\psi(\mathcal{C})| \\ &\leq \varepsilon_r |\psi(\mathcal{C})|. \end{aligned}$$

Applying the above upper bound to Ineq. (6) completes the proof of the claim.  $\triangleleft$

To prove Ineq. (4), it remains to lower bound  $|\tilde{\psi}(\hat{\mathcal{B}})|$ , in the inequality of Claim 4.6, in terms of  $|\psi(\mathcal{C})|$ . Below,  $|\psi(\hat{\mathcal{C}})| = |\psi(\mathcal{C})|$  holds since every  $S \in \mathcal{C}$  is allocated to some  $\mathcal{G}_i \in \hat{\mathcal{C}}$ .

$$\begin{aligned} |\tilde{\psi}(\mathcal{B})| &\geq |\tilde{\psi}(\hat{\mathcal{B}})| - \varepsilon_r |\psi(\mathcal{C})| && \text{Claim 4.6} \\ &\geq \frac{|\psi(\hat{\mathcal{C}})|}{H_{\hat{\ell}}} - \varepsilon_r |\psi(\mathcal{C})| && \text{Line 8 and Theorem 4.1} \\ &= \frac{|\psi(\hat{\mathcal{C}})|}{H_{\lceil r(r-1)/\varepsilon_r \rceil}} - \varepsilon_r |\psi(\mathcal{C})| && \text{value of } \hat{\ell} \text{ (Line 1)} \\ &= \frac{|\psi(\mathcal{C})|}{H_{\lceil r(r-1)/\varepsilon_r \rceil}} - \varepsilon_r |\psi(\mathcal{C})| && |\psi(\hat{\mathcal{C}})| = |\psi(\mathcal{C})| \\ &= \left( \frac{1}{H_{\lceil r(r-1)/\varepsilon_r \rceil}} - \varepsilon_r \right) |\psi(\mathcal{C})|. \end{aligned}$$

**Ineq. (5).** It remains to show that there exists a choice of  $\varepsilon_r$  that implies Ineq. (5).

$\triangleright$  **Claim 4.7 (\*).** Setting  $\varepsilon_r = (9.27 \ln r)^{-1} (2 \ln r + 2 \ln \ln r + 5.61)^{-1}$  implies Ineq. (5).

This completes the proof of Theorem 4.4.  $\blacktriangleleft$

### 4.3 UniqueGreedySize

In this section, we present `UNIQUEGREEDYSIZE`, with pseudocode in Algorithm 3, derived by combining `UNIQUEGREEDYFREQ` with the approach in Theorem 4.2 of Demaine et al. [9]. The purpose of this algorithm is to take a collection,  $\mathcal{C}$ , with maximum set size  $d$ , an error parameter,  $\varepsilon_d \in (0, 1)$ , and another error parameter,  $\hat{\varepsilon}_d \in (0, 1)$ , and return a  $\mathcal{B} \subseteq \mathcal{C}$  whose unique coverage is at least a logarithmic factor of  $\mathcal{C}$ 's coverage, where the factor depends on  $d$ ,  $\varepsilon_d$ , and  $\hat{\varepsilon}_d$ . We state this formally in Theorem 4.8 and give the proof for completeness; in fact, our proof slightly generalizes the proof of Theorem 4.2 of Demaine et al. [9], by allowing an arbitrary  $\varepsilon_d$  rather than fixing  $\varepsilon_d = 1/2$ .

**UniqueGreedySize Overview.** `UNIQUEGREEDYSIZE` first modifies  $\mathcal{C}$  into a “minimal” collection by discarding each set  $T$  that uniquely covers no element. Then it checks if  $\mathcal{C}$ 's size is at least an  $\varepsilon_d$  factor of its own coverage. If so, then it assigns  $\mathcal{C}$  to the solution  $\mathcal{B}$ . Otherwise, it constructs a sub-instance on those elements of frequency at most  $d$  and calls `UNIQUEGREEDYFREQ` on the sub-instance with error  $\hat{\varepsilon}_d$  to get  $\hat{\mathcal{B}}$ . Returned solution  $\mathcal{B}$  comprises each set  $S \in \mathcal{C}$  whose intersection with  $\hat{U}$  was selected into  $\hat{\mathcal{B}}$ .

$\blacktriangleright$  **Theorem 4.8.** *Let  $\mathcal{C}$  denote a collection of sets,  $d$  denote the maximum size of a set in  $\mathcal{C}$ ,  $\varepsilon_d \in (0, 1)$  denote an error parameter, and  $\hat{\varepsilon}_d \in (0, 1)$  denote an error parameter passed to `UNIQUEGREEDYFREQ`. Then `UNIQUEGREEDYSIZE` returns a collection  $\mathcal{B} \subseteq \mathcal{C}$  satisfying*

$$|\tilde{\psi}(\mathcal{B})| \geq \min(\varepsilon_d, (1 - \varepsilon_d)\beta(d, \hat{\varepsilon}_d)) |\psi(\mathcal{C})|, \quad (9)$$

■ **Algorithm 3** UNIQUEGREEDYSIZE.

---

**Input:**  $\mathcal{C}$ : collection with maximum set size  $d$ ,  $\varepsilon_d \in (0, 1)$ : error parameter,  
 $\hat{\varepsilon}_d \in (0, 1)$ : error parameter used in UNIQUEGREEDYFREQ.  
**Output:**  $\mathcal{B} \subseteq \mathcal{C}$ : subcollection satisfying  $|\tilde{\psi}(\mathcal{B})| \geq \min(\varepsilon_d, (1 - \varepsilon_d)\beta(d, \hat{\varepsilon}_d))|\psi(\mathcal{C})|$   
 where  $\beta(d, \hat{\varepsilon}_d) = 1/H_{\lceil d(d-1)/\hat{\varepsilon}_d \rceil} - \hat{\varepsilon}_d$ .

```

1 while  $T \leftarrow \arg \min_{S \in \mathcal{C}} |S \cap \tilde{\psi}(\mathcal{C})|$  satisfies  $|T \cap \tilde{\psi}(\mathcal{C})| = 0$  do // Make  $\mathcal{C}$  minimal
2    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{T\}$ 
3 if  $|\mathcal{C}| \geq \varepsilon_d |\psi(\mathcal{C})|$  then
4    $\mathcal{B} \leftarrow \mathcal{C}$ 
5 else // Define instance on elements with freq.  $\leq d$ 
6    $\hat{U} \leftarrow \{x \in \psi(\mathcal{C}) : \text{freq}_{\mathcal{C}}(x) \leq d\}$ 
7    $\hat{\mathcal{C}} \leftarrow \{S \cap \hat{U} : S \in \mathcal{C}\}$ 
8    $\hat{\mathcal{B}} \leftarrow \text{UNIQUEGREEDYFREQ}(\hat{\mathcal{C}}, \hat{\varepsilon}_d)$ 
9    $\mathcal{B} \leftarrow \emptyset$ 
10  for  $S \cap \hat{U} \in \hat{\mathcal{B}}$  do // Construct returned solution
11     $\mathcal{B} \leftarrow \mathcal{B} \cup \{S\}$ 
12 return  $\mathcal{B}$ 
    
```

---

where  $\beta(d, \hat{\varepsilon}_d) = 1/(H_{\lceil d(d-1)/\hat{\varepsilon}_d \rceil}) - \hat{\varepsilon}_d$  denotes the unique coverage ratio of UNIQUEGREEDYFREQ. Moreover, by assigning  $\varepsilon_d = (1/\beta(d, \hat{\varepsilon}_d) + 1)^{-1}$ ,  $\hat{\varepsilon}_d = (c_1 \ln d)^{-1} (2 \ln d + 2 \ln \ln d + c_2)^{-1}$ , and appropriate constants to  $c_1$  and  $c_2$ , we derive from Ineq. (9) the simpler inequality below.

$$|\tilde{\psi}(\mathcal{B})| \geq \frac{1}{2 \ln d + o(\log d)} |\psi(\mathcal{C})|. \quad (10)$$

**Proof.** We begin by proving Ineq. (9). Discarding sets from  $\mathcal{C}$  that uniquely cover no elements, as in Lines 1–2, does not affect  $\psi(\mathcal{C})$ . So assume that  $\mathcal{C}$  is minimal, i.e., every  $S \in \mathcal{C}$  uniquely covers at least one element. This means that  $|\tilde{\psi}(\mathcal{C})| \geq |\mathcal{C}|$ .

Now one of two cases must hold: (i)  $|\mathcal{C}| \geq \varepsilon_d |\psi(\mathcal{C})|$ ; or (ii)  $|\mathcal{C}| < \varepsilon_d |\psi(\mathcal{C})|$ . The final ratio in Ineq. (9) is the minimum ratio achieved out of these two cases.

In case (i), UNIQUEGREEDYSIZE returns the solution  $\mathcal{B} = \mathcal{C}$ , by the success of the condition in Line 3. Further,  $|\tilde{\psi}(\mathcal{B})| = |\tilde{\psi}(\mathcal{C})| \geq |\mathcal{C}| \geq \varepsilon_d |\psi(\mathcal{C})|$  holds by the minimality of  $\mathcal{C}$ . Thus,  $\mathcal{B}$  satisfies Ineq. (9) in case (i).

In case (ii), we show that the set  $\hat{U}$  of elements  $x \in \psi(\mathcal{C})$  with  $\text{freq}_{\mathcal{C}}(x) \leq d$ , as in Line 6, satisfies  $|\hat{U}| \geq (1 - \varepsilon_d) |\psi(\mathcal{C})|$ . We have

$$\begin{aligned}
 |U \setminus \hat{U}| &< \frac{1}{d} \sum_{S \in \mathcal{C}} |S| && \text{for all } x \in U \setminus \hat{U}: \text{freq}_{\mathcal{C}}(x) > d \\
 &\leq |\mathcal{C}| && \text{max. set size is } d \\
 &< \varepsilon_d |\psi(\mathcal{C})|, && \text{case (ii)} \\
 |\hat{U}| &> (1 - \varepsilon_d) |\psi(\mathcal{C})|.
 \end{aligned}$$

By the Line 7 definition,  $\psi(\hat{\mathcal{C}}) = \hat{U}$ , so  $|\psi(\hat{\mathcal{C}})| \geq (1 - \varepsilon_d) |\psi(\mathcal{C})|$ . Therefore, calling UNIQUEGREEDYFREQ on  $\hat{\mathcal{C}}$  with maximum frequency  $d$  and error  $\hat{\varepsilon}_d$ , as in Line 8, gives a collection  $\hat{\mathcal{B}}$  satisfying  $|\tilde{\psi}(\hat{\mathcal{B}})| \geq \beta(d, \hat{\varepsilon}_d) |\psi(\hat{\mathcal{C}})| \geq \beta(d, \hat{\varepsilon}_d) (1 - \varepsilon_d) |\psi(\mathcal{C})|$ . Likewise, by definition, in Lines 9–11,  $|\tilde{\psi}(\mathcal{B})| \geq |\tilde{\psi}(\hat{\mathcal{B}})|$ . Thus,  $\mathcal{B}$  satisfies Ineq. (9) in case (2), proving Theorem 4.8.

**Ineq. (10).** We first maximize  $\min(\varepsilon_d, (1 - \varepsilon_d)\beta(d, \hat{\varepsilon}_d))$  with respect to  $\varepsilon_d$  by setting the two arguments as equal; this makes the RHS of Ineq. (9) equal to  $\varepsilon_d = (1/\beta(d, \hat{\varepsilon}_d) + 1)^{-1}$ . Then, by assigning  $\hat{\varepsilon}_d = (c_1 \ln d)^{-1}(2 \ln d + 2 \ln \ln d + c_2)^{-1}$  and appropriate constants to  $c_1$  and  $c_2$ , **UNIQUEGREEDYFREQ**'s unique coverage ratio satisfies  $\beta(d, \hat{\varepsilon}_d) \geq 1/(2 \ln d + o(\log d))$  as in Theorem 4.4. Further substituting this into the RHS of Ineq. (9) proves Ineq. (10). This completes the proof of Theorem 4.8. ◀

## 5 Space Lower Bound for a $(1.5 + o(1)) / (\ln k - 1)$ -Approximation

In this section, we prove the following theorem:

► **Theorem 5.1.** *Let  $e^{2.5} \leq k \leq m$ ,  $a = k \ln m + \ln(k/0.05)$ , and assume the universe size to be  $n = k(k-1) \sum_{t=1}^k \lceil a/t \rceil$ . Then every constant-pass randomized streaming algorithm for **Max Unique Coverage** that, with probability at least 0.95, has an approximation factor of  $(3/2 + 3/\sqrt{2k}) / (H_k - 1)$ , requires  $\Omega(m/k^2)$  space.*

### 5.1 High-Level Ideas of the Reduction

Similar to other approaches [16, 15], we prove our space lower bound by reducing the problem of  $k$ -player Set Disjointness (with the unique intersection promise) in the one-way communication model, denoted by **Disj**, to **Max Unique Coverage** in the stream model.

**Set Disjointness in the One-Way Communication Model.** In the one-way communication model, players must take turns in some fixed order to send a message to the player next in order, i.e., the  $j^{\text{th}}$  player can only send a message to the  $(j+1)^{\text{th}}$  player. There can be  $p \geq 1$  rounds of communication, where a single round is completed once every player has taken their turn. The last player can send a message back to the 1<sup>st</sup> player at the end of a round if there is a next round.

In an instance of **Disj**, each player  $j \in [k]$  is given a set of integers  $D_j \subseteq [m]$ . Moreover, it is promised that only two kinds of instances can occur:

**NO instance.** All sets  $D_j$  are pairwise disjoint.

**YES instance.** There is a unique integer  $i^* \in [m]$  such that, for all  $j \in [k]$ ,  $i^* \in D_j$ .

The goal then is for the  $k^{\text{th}}$  player (in the final round) to correctly answer, with probability at least 0.9, whether the given sets form a YES or NO instance.

The communication complexity of **Disj** in the  $p$ -round one-way communication model is  $\Omega(m/k)$ , even for randomized protocols and even when the players can use public randomness [5]. Thus, as there are  $\leq pk$  messages, every (randomized) protocol for **Disj** must have at least one message of size  $\Omega(m/(pk^2))$  in the worst case.

**Reduction Overview.** Given an instance of **Disj**, the main goal of the reduction, with parameter  $a$ , is for the players to construct an instance of **Max Unique Coverage** in a stream such that if they were given a NO instance of **Disj**, the optimal unique coverage is less than  $ak^2(1.5 + o(1))$  (with high probability); whereas if the players were given a YES instance of **Disj**, the optimal unique coverage is at least  $ak^2(H_k - 1)$ . The ratio of these optimal unique coverages is less than  $(1.5 + o(1)) / (H_k - 1)$ , so the players can use a  $(1.5 + o(1)) / (H_k - 1)$ -approximation streaming algorithm on the **Max Unique Coverage** instance to distinguish between a NO and YES instance. By a standard argument, this implies a protocol for **Disj** which involves each player sending the memory of the streaming algorithm in a message to

the next player. A constant-pass  $O(s)$ -space streaming algorithm implies a protocol with a maximum message size of  $O(s)$  in constant rounds of communication where each pass of the streaming algorithm takes one round. Thus, a  $(1.5 + o(1))/(H_k - 1)$ -approximation streaming algorithm for **Max Unique Coverage** requires  $\Omega(m/k^2)$  space.

**Intuition of Max Unique Coverage Construction.** Here, we give the intuition for constructing the streaming instance of **Max Unique Coverage** that achieves the optimal unique coverages above, with details in the proof of Theorem 5.1.

Let the universe of the **Max Unique Coverage** instance be  $U = U_1 \cup \dots \cup U_k$ , where  $U_1, \dots, U_k$  are  $k$  disjoint sub-universes such that  $|U_t| = k(k-1)\lceil a/t \rceil$  (for a sufficiently large  $a$  as in Theorem 5.1). Then, for each  $i \in [m]$ , each player  $j$  constructs  $S_j^i \subseteq U$  such that  $S_1^i, \dots, S_k^i$  satisfy the following properties:

1. Each set  $S_j^i$  covers  $t/k$  proportion of  $U_t$  for all  $t$ .
2. For each  $t \in [k]$ , the sets  $S_1^i, \dots, S_k^i$  partition a proportion,  $q_t \in [0, 1]$ , of  $U_t$  while having a common intersection in the remaining  $(1 - q_t)$  proportion of  $U_t$ . I.e., sets with identical  $i$  form a “sunflower”, with their overlap concentrated in the sunflower’s “kernel”.
3. The choice of elements to be covered by  $S_j^i$  are independent and uniform random with respect to  $i \in [m]$ .

The above construction ensures that (with high probability) every collection of  $\ell \in [k]$  sets,  $S_{j_1}^{i_1}, \dots, S_{j_\ell}^{i_\ell}$ , with distinct  $i_1, \dots, i_\ell$  has a unique coverage less than  $ak^2(1.5 + o(1))$  (with high probability); whereas a collection of  $\ell = k$  sets with identical  $i_1, \dots, i_\ell$  has a unique coverage of at least  $ak^2(H_k - 1)$ . Observe that  $k \geq e^{2.5}$  ensures that  $H_k - 1 > 1.5 + o(1)$ .

Finally, to construct the streaming instance of **Max Unique Coverage**, each player  $j$  inserts  $S_j^i$  into the stream iff  $i \in D_j$ . This means that, given a NO instance, every set  $S_j^i$  in the stream has a distinct  $i$ ; whereas given a YES instance, there exists a collection of  $\ell = k$  sets in the stream all indexed by  $i^*$ , the unique integer contained in all  $D_1, \dots, D_k$ . This results in the optimal unique coverages for the NO and YES instances as required.

## 5.2 Proof of Theorem 5.1

We show a reduction from **Disj** to **Max Unique Coverage**. Assume without loss of generality that the sets  $D_j$  are padded so that  $|D_1 \cup \dots \cup D_k| \geq m/4 \geq m/k^2$  holds for  $k \geq 2$ .

**Construction of Max Unique Coverage Instance.** First, the players define the **Max Unique Coverage** universe as  $U = U_1 \cup \dots \cup U_k$ , where  $U_1, \dots, U_k$  are  $k$  disjoint sub-universes such that  $|U_t| = k(k-1)\lceil a/t \rceil$ . Observe that, as per the assumption in Theorem 5.1, we have  $n = |U| = \sum_{t=1}^k |U_t| = k(k-1) \sum_{t=1}^k \lceil a/t \rceil$ .

The players now construct the **Max Unique Coverage** sets so that they satisfy the properties given in the overview. For each  $i \in [m]$  and  $t \in [k]$ , the players define  $\tilde{U}_t^i \subseteq U_t$  as an independent and uniformly chosen random subset of size  $q_t = (k-t)/(k-1)$  proportion of  $U_t$ ; they then independently and uniformly-at-random partition  $\tilde{U}_t^i$  into  $k$  equally sized sets,  $P_{t,1}^i, \dots, P_{t,k}^i$ ; the players agree on all of these choices using public randomness. For example, the players obtain a common random permutation of  $U_t$  and pick the corresponding parts in order. Note that  $\tilde{U}_t^i$  can be divided into  $k$  equal sets since  $|\tilde{U}_t^i|/k$  is an integer, viz.

$$\frac{|\tilde{U}_t^i|}{k} = \frac{q_t |U_t|}{k} = \frac{(k-t)k(k-1)}{k(k-1)} \left\lceil \frac{a}{t} \right\rceil = (k-t) \left\lceil \frac{a}{t} \right\rceil.$$

Then, for each  $i \in [m]$ , each player  $j$  defines their set  $S_j^i$  such that, for each  $t \in [k]$ , it covers the  $j^{\text{th}}$  set in the partition of  $\tilde{U}_t^i$ , namely  $P_{t,j}^i$ ; and it covers all of  $U_t \setminus \tilde{U}_t^i$ . More precisely,

$$S_j^i = \bigcup_{t=1}^k [P_{t,j}^i \cup (U_t \setminus \tilde{U}_t^i)].$$

Observe Claim 5.2, which we use in Claim 5.4 later.

▷ **Claim 5.2.** For each  $i \in [m]$ ,  $j \in [k]$ , and  $t \in [k]$ ,  $S_j^i$  covers  $t/k$  proportion of  $U_t$ .

Proof. The proportion of  $U_t$  that  $S_j^i$  covers is  $|S_j^i \cap U_t|/|U_t|$ , which we prove to be  $t/k$  below.

$$\begin{aligned} \frac{|S_j^i \cap U_t|}{|U_t|} &= \frac{|P_{t,j}^i|}{|U_t|} + \frac{|U_t \setminus \tilde{U}_t^i|}{|U_t|} = \frac{|\tilde{U}_t^i|}{k|U_t|} + \frac{|U_t \setminus \tilde{U}_t^i|}{|U_t|} = \frac{q_t}{k} + 1 - q_t \\ &= \frac{k-t}{k(k-1)} + 1 - \frac{k-t}{k-1} = \frac{k-t}{k(k-1)} + \frac{t-1}{k-1} \\ &= \frac{k-t+kt-k}{k(k-1)} = \frac{kt-t}{k(k-1)} = \frac{t(k-1)}{k(k-1)} = \frac{t}{k}. \quad \triangleleft \end{aligned}$$

To complete the construction, each player  $j$  inserts set  $S_j^i$  into the stream iff  $i \in D_j$ . There are  $\Theta(m)$  sets inserted into the stream since  $m/4 \leq |D_1 \cup \dots \cup D_k| \leq m$ .

**Upper Bound on Optimal Unique Coverage in a NO Instance.** Next, we prove Lemma 5.3, which implies the required upper bound on the optimal unique coverage in a NO instance. We say that a collection  $\mathcal{L}_{\text{di}} = \{S_{j_1}^{i_1}, \dots, S_{j_\ell}^{i_\ell}\}$  with distinct  $i_1, \dots, i_\ell$  is a *player-distinct collection*; we also say that  $\mathcal{L}_{\text{di}}$  is *feasible* if it contains at most  $k$  sets. Note that in the **Max Unique Coverage** instance generated from a NO instance of **Disj**, every feasible solution is a player-distinct collection. Thus, it suffices to upper bound the unique coverage of every feasible player-distinct collection.

► **Lemma 5.3.** *With probability at least 0.95, every feasible player-distinct collection  $\mathcal{L}_{\text{di}}$  satisfies  $|\tilde{\psi}(\mathcal{L}_{\text{di}})| < ak^2(3/2 + 3/\sqrt{2k})$ .*

**Proof.** First, we upper bound  $\mathbb{E}[|\tilde{\psi}(\mathcal{L}_{\text{di}}) \cap U_t|]$  for every feasible player-distinct collection,  $\mathcal{L}_{\text{di}}$ , and for every sub-universe  $U_t$  (Claim 5.4), then we use Hoeffding's inequality to prove an upper bound on  $|\tilde{\psi}(\mathcal{L}_{\text{di}}) \cap U_t|$  that with high probability, holds simultaneously for every  $\mathcal{L}_{\text{di}}$  and  $U_t$  (Claim 5.5). Summing the bound in Claim 5.5 over all  $k$  sub-universes suffices.

For a feasible player-distinct collection  $\mathcal{L}_{\text{di}}$ , let  $X_{x, \mathcal{L}_{\text{di}}}$  be the random variable such that  $X_{x, \mathcal{L}_{\text{di}}} = 1$  if element  $x \in \tilde{\psi}(\mathcal{L}_{\text{di}})$ , and  $X_{x, \mathcal{L}_{\text{di}}} = 0$  otherwise. This means that for each sub-universe  $U_t$ , we have

$$|\tilde{\psi}(\mathcal{L}_{\text{di}}) \cap U_t| = \sum_{x \in U_t} X_{x, \mathcal{L}_{\text{di}}}; \text{ and so } |\tilde{\psi}(\mathcal{L}_{\text{di}})| = \sum_{t=1}^k \sum_{x \in U_t} X_{x, \mathcal{L}_{\text{di}}}. \quad (11)$$

▷ **Claim 5.4 (\*)**. For each feasible player-distinct  $\mathcal{L}_{\text{di}}$  of  $\ell \in [k]$  sets and each sub-universe  $U_t$ , it holds that  $\mathbb{E}[|\tilde{\psi}(\mathcal{L}_{\text{di}}) \cap U_t|] \leq k(a+t)\ell(1-t/k)^{\ell-1}$ .

▷ **Claim 5.5 (\*)**. With probability at least 0.95, for every feasible player-distinct  $\mathcal{L}_{\text{di}}$  of  $\ell \in [k]$  sets and every sub-universe  $U_t$ , it holds that

$$|\tilde{\psi}(\mathcal{L}_{\text{di}}) \cap U_t| < k(a+t)\ell \left(1 - \frac{t}{k}\right)^{\ell-1} + \frac{k(a+t)}{(2t)^{1/2}}.$$

## 25:20 Maximum Unique Coverage on Streams

Finally, summing the inequality of Claim 5.5 over the  $k$  sub-universes gives an upper bound on  $|\tilde{\psi}(\mathcal{L}_{\text{di}})|$  that holds simultaneously for every feasible player-distinct collection  $\mathcal{L}_{\text{di}}$  with high probability. We finalize the proof of Lemma 5.3 in Claim 5.6.

▷ Claim 5.6 (\*). With probability at least 0.95,  $|\tilde{\psi}(\mathcal{L}_{\text{di}})| < ak^2(3/2 + 3/\sqrt{2k})$ . ◀

**Lower Bound on Optimal Unique Coverage in a YES Instance.** Lemma 5.7 supports the required lower bound on the optimal unique coverage in a YES instance.

► **Lemma 5.7.** For all  $i$ , collection  $\mathcal{L}_{\text{id}} = \{S_1^i, \dots, S_k^i\}$  satisfies  $|\tilde{\psi}(\mathcal{L}_{\text{id}})| \geq ak^2(H_k - 1)$ .

**Proof.** For each  $t \in [k]$ ,  $\mathcal{L}_{\text{id}}$  uniquely covers  $|\tilde{U}_t^i|$  by construction. Below, the inequality holds since  $|U_t| = k(k-1)\lceil a/t \rceil \geq ak(k-1)/t$ .

$$\begin{aligned} |\tilde{\psi}(\mathcal{L}_{\text{id}})| &= \sum_{t=1}^k |\tilde{U}_t^i| = \sum_{t=1}^k q_t |U_t| \geq \sum_{t=1}^k \frac{k-t}{k-1} \frac{ak(k-1)}{t} = \sum_{t=1}^k \frac{k-t}{t} ak \\ &= ak \sum_{t=1}^k \left( \frac{k}{t} - 1 \right) = ak \left( k \sum_{t=1}^k \frac{1}{t} - k \right) = ak^2(H_k - 1). \end{aligned} \quad \blacktriangleleft$$

To conclude, when the players reduce from a NO instance of **Disj**, with probability at least 0.95, the optimal unique coverage is less than  $ak^2(3/2 + 3/\sqrt{2k})$ , since the streamed sets are player distinct and by Lemma 5.3; whereas when they reduce from a YES instance, the optimal unique coverage is at least  $ak^2(H_k - 1)$  since the sets  $S_1^{i^*}, \dots, S_k^{i^*}$  are in the stream and by Lemma 5.7. The required optimal unique coverage in a NO instance fails with probability at most 0.05. Let  $\alpha = (3/2 + 3/\sqrt{2k})/(H_k - 1)$ . Given a randomized  $O(s)$ -space  $\alpha$ -approximation streaming algorithm with failure probability at most 0.05, the players can run this algorithm on the **Max Unique Coverage** instance to distinguish between a NO or YES instance with failure probability at most 0.1. This implies a protocol for **Disj** with maximum message size  $O(s)$ . Thus, a constant-pass randomized  $\alpha$ -approximation streaming algorithm with success probability at least 0.95 requires  $\Omega(m/k^2)$  space.

## 6 Subsampling for the Data Stream

Here we outline the subsampling approach from [15]. Given a data stream instance of **Max Unique Coverage**, it is possible to construct a number of *subsamped* instances by sampling the universe  $U$  at varying rates. By running an algorithm on these subsampled instances in parallel, we lose only a small error in approximation w.h.p. while only needing to store sets of size  $O(k \log m/\varepsilon^2)$ . We summarize the overall approach in Lemma 2.1 and give a proof sketch.

**Proof Sketch of Lemma 2.1.** Given an instance of **Max Unique Coverage** with universe  $U$  and collection of sets  $\mathcal{V}$ , let  $v$  be a guess of the optimal solution value; each subsampled instance corresponds to some value of  $v$  (we calculate these guesses shortly). Let  $h : U \rightarrow \{0, 1\}$  be a hash function that is  $\Omega(k \log m/\varepsilon^2)$ -wise independent such that

$$\Pr[h(x) = 1] = p = \frac{ck \log m}{\varepsilon^2 v},$$

where  $c$  is a sufficiently large constant. Let  $U' = \{x \in U : h(x) = 1\}$  be the subsampled universe,  $S' = S \cap U'$ ,  $\mathcal{V}' = \{S' : S \in \mathcal{V}\}$  be the subsampled subsets, and  $\text{OPT}'$  be the optimal unique coverage in the subsampled instance. Further, let  $\mathcal{B}'$  be a solution from  $\mathcal{V}'$



and  $\mathcal{B}$  be the corresponding solution from the original collection  $\mathcal{V}$ . Then Lemma 6.1 below (a restatement of [15, Lemma 23]) shows that, in a subsampled instance where  $v \leq \text{OPT}$ , w.h.p., the loss in approximation is at most  $2\varepsilon$ .

► **Lemma 6.1** ([15, Lemma 23]). *If  $v \leq \text{OPT}$ , then with probability at least  $1 - 1/\text{poly}(m)$ , we have that*

$$(1 + \varepsilon)p\text{OPT} \geq \text{OPT}' \geq (1 - \varepsilon)p\text{OPT}.$$

Furthermore, for some  $\alpha \in (0, 1)$ , if  $\mathcal{B}' \subseteq \mathcal{V}'$  satisfies  $|\tilde{\psi}(\mathcal{B}')| \geq \alpha(1 - \varepsilon)p\text{OPT}$ , then  $|\tilde{\psi}(\mathcal{B})| \geq (\alpha - 2\varepsilon)\text{OPT}$ .

We guess  $v = 2^i$  for each  $i \in [\lceil \log_2 n \rceil]$  and construct a subsampled instance for each  $v$  in parallel. Then, in the particular subsampled instance where  $\text{OPT}/2 \leq v \leq \text{OPT}$ , Lemma 6.1 implies the following upper bound on every set size  $|S'|$  with probability  $1 - 1/\text{poly}(m)$ .

$$|S'| \leq \text{OPT}' \leq (1 + \varepsilon)p\text{OPT} = (1 + \varepsilon) \frac{ck \log m}{\varepsilon^2 v} \text{OPT} \leq (1 + \varepsilon) \frac{2ck \log m}{\varepsilon^2} = O\left(\frac{k \log m}{\varepsilon^2}\right).$$

To ensure that we only ever store sets of size  $O(k \log m / \varepsilon^2)$ , we terminate every subsampled instance that contains a set  $S'$  with  $|S'| > (2ck \log m / \varepsilon^2)(1 + \varepsilon)$ . W.h.p., this does not terminate the subsampled instance where  $\text{OPT}/2 \leq v \leq \text{OPT}$  by the above upper bound on  $|S'|$  for every  $S'$  in this particular instance.

This means that, out of the nonterminated subsampled instances, we should select the one with the smallest  $v$  and return the corresponding solution, giving an  $(\alpha - 2\varepsilon)$ -approximation for the original instance w.h.p. (this works even if the smallest nonterminated guess satisfies  $v < \text{OPT}/2$  since Lemma 6.1 holds for all  $v \leq \text{OPT}$ ).

The overall space complexity,  $\lceil \log_2 n \rceil \cdot s \cdot O(k \log m \log n / \varepsilon^2)$ , follows from the number of guesses of  $v$  and, for each guess, the algorithm storing at most  $s$  sets of size  $O(k \log m / \varepsilon^2)$  and using  $O(\log n)$  bits to store each element. ◀

## 7 Conclusions

We are pleased to present a suite of algorithms, and a streaming lower bound, for **Max Unique Coverage**. The component algorithms that build a solution to **Max Unique Coverage** from a solution to **Max Coverage** serve to support a fixed-parameter tractable approximation scheme (FPT-AS). The lower bound shows that  $\Omega(m/k^2)$  space is required even to get within a  $(1.5 + o(1))/(\ln k - 1)$  factor of optimal.

A plausible future direction would be to reduce, or indeed eliminate, the role of the upper bound on the unique coverage ratio,  $\phi$ , in the kernel size in a FPT-AS. This would match the kernel size used in existing FPT-ASs for **Max Coverage**, but may not be possible due to the inherent hardness of **Max Unique Coverage**. Another direction would be proving a streaming lower bound with a tighter approximation threshold. This may require a reduction from a different communication problem, rather than the renowned  $k$ -player Set Disjointness.

---

## References

- 1 Sepehr Assadi. Tight space-approximation tradeoff for the multi-pass streaming set cover problem. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 321–335, 2017.
- 2 Giorgio Ausiello, Nicolas Boria, Aristotelis Giannakos, Giorgio Lucarelli, and V. Th Paschos. Online maximum  $k$ -coverage. *Discrete Applied Mathematics*, 160(13-14):1901–1913, 2012.

- 3 MohammadHossein Bateni, Hossein Esfandiari, and Vahab S. Mirrokni. Almost optimal streaming algorithms for coverage problems. In Christian Scheideler and MohammadTaghi Hajiaghayi, editors, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 13–23. ACM, 2017. doi:10.1145/3087556.3087585.
- 4 Édouard Bonnet, Vangelis Th Paschos, and Florian Sikora. Parameterized exact and approximation algorithms for maximum  $k$ -set cover and related satisfiability problems. *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications*, 50(3):227–240, 2016.
- 5 Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.*, pages 107–117. IEEE, 2003.
- 6 Rajesh Chitnis and Graham Cormode. Towards a theory of parameterized streaming algorithms. In *14th International Symposium on Parameterized and Exact Computation*, 2019.
- 7 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*, pages 1326–1344. SIAM, 2016.
- 8 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. New streaming algorithms for parameterized maximal matching & beyond. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, pages 56–58, 2015.
- 9 Erik D. Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad R. Salavatipour. Combination can be hard: Approximability of the unique coverage problem. *SIAM Journal on Computing*, 38(4):1464–1483, 2008.
- 10 Erik D. Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. On streaming and communication complexity of the set cover problem. In Fabian Kuhn, editor, *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, volume 8784 of *Lecture Notes in Computer Science*, pages 484–498. Springer, 2014. doi:10.1007/978-3-662-45174-8\_33.
- 11 Venkatesan Guruswami and Euiwoong Lee. Nearly optimal NP-hardness of unique coverage. *SIAM Journal on Computing*, 46(3):1018–1028, 2017.
- 12 Sarel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards tight bounds for the streaming set cover problem. In *PODS*, pages 371–383. ACM, 2016.
- 13 Chien-Chung Huang and François Sellier. Matroid-constrained maximum vertex cover: Approximate kernels and streaming algorithms. In *SWAT 2022*, 2022.
- 14 Pasin Manurangsi. A note on max  $k$ -vertex cover: Faster FPT-AS, smaller approximate kernel and improved approximation. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 15 Andrew McGregor, David Tench, and Hoa T. Vu. Maximum Coverage in the Data Stream Model: Parameterized and Generalized. In *24th International Conference on Database Theory*, 2021.
- 16 Andrew McGregor and Hoa T. Vu. Better streaming algorithms for the maximum coverage problem. *Theory of Computing Systems*, 63(7):1595–1619, 2019.
- 17 Neeldhara Misra, Hannes Moser, Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. The parameterized complexity of unique coverage and its variants. *Algorithmica*, 65:517–544, 2013.
- 18 Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proceedings of the 2009 siam international conference on data mining*, pages 697–708. SIAM, 2009.
- 19 François Sellier. Parameterized matroid-constrained maximum coverage. In *31st Annual European Symposium on Algorithms (ESA 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

- 20 Piotr Skowron. FPT approximation schemes for maximizing submodular functions. *Information and Computation*, 257:65–78, 2017. doi:10.1016/j.ic.2017.10.002.
- 21 Piotr Skowron and Piotr Faliszewski. Fully proportional representation with approval ballots: Approximating the MaxCover problem with bounded frequencies in FPT time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2124–2130, 2015.
- 22 Huiwen Yu and Dayu Yuan. Set coverage problems in a one-pass data stream. In *Proceedings of the 2013 SIAM international conference on data mining*, pages 758–766. SIAM, 2013.



# Improved Streaming Algorithm for the Klee’s Measure Problem and Generalizations \*

Mridul Nandi (R) ✉

Indian Statistical Institute, Kolkata, India

N. V. Vinodchandran (R) ✉

University of Nebraska, Lincoln, USA

Arijit Ghosh (R) ✉

Indian Statistical Institute, Kolkata, India

Kuldeep S. Meel (R) ✉

University of Toronto, Canada

Soumit Pal (R) ✉

Indian Statistical Institute, Kolkata, India

Sourav Chakraborty ✉

Indian Statistical Institute, Kolkata, India

---

## Abstract

Estimating the size of the union of a stream of sets  $S_1, S_2, \dots, S_M$  where each set is a subset of a known universe  $\Omega$  is a fundamental problem in data streaming. This problem naturally generalizes the well-studied  $F_0$  estimation problem in the streaming literature, where each set contains a single element from the universe. We consider the general case when the sets  $S_i$  can be succinctly represented and allow efficient membership, cardinality, and sampling queries (called a Delphic family of sets). A notable example in this framework is the Klee’s Measure Problem (KMP), where every set  $S_i$  is an axis-parallel rectangle in  $d$ -dimensional spaces ( $\Omega = [\Delta]^d$  where  $[\Delta] := \{1, \dots, \Delta\}$  and  $\Delta \in \mathbb{N}$ ). Recently, Meel, Chakraborty, and Vinodchandran (PODS-21, PODS-22) designed a streaming algorithm for  $(\epsilon, \delta)$ -estimation of the size of the union of set streams over Delphic family with space and update time complexity  $O\left(\frac{\log^3 |\Omega|}{\epsilon^2} \cdot \log \frac{1}{\delta}\right)$  and  $\tilde{O}\left(\frac{\log^4 |\Omega|}{\epsilon^2} \cdot \log \frac{1}{\delta}\right)$ , respectively.

This work presents a new, sampling-based algorithm for estimating the size of the union of Delphic sets that has space and update time complexity  $\tilde{O}\left(\frac{\log^2 |\Omega|}{\epsilon^2} \cdot \log \frac{1}{\delta}\right)$ . This improves the space complexity bound by a  $\log |\Omega|$  factor and update time complexity bound by a  $\log^2 |\Omega|$  factor.

A critical question is whether quadratic dependence of  $\log |\Omega|$  on space and update time complexities is necessary. Specifically, can we design a streaming algorithm for estimating the size of the union of sets over Delphic family with space and complexity linear in  $\log |\Omega|$  and update time  $\text{poly}(\log |\Omega|)$ ? While this appears technically challenging, we show that establishing a lower bound of  $\omega(\log |\Omega|)$  with  $\text{poly}(\log |\Omega|)$  update time is beyond the reach of current techniques. Specifically, we show that under certain hard-to-prove computational complexity hypothesis, there is a streaming algorithm for the problem with optimal space complexity  $O(\log |\Omega|)$  and update time  $\text{poly}(\log(|\Omega|))$ . Thus, establishing a space lower bound of  $\omega(\log |\Omega|)$  will lead to break-through complexity class separation results.

**2012 ACM Subject Classification** Theory of computation → Sketching and sampling

**Keywords and phrases** Sampling, Streaming, Klee’s Measure Problem

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.26

**Category** APPROX

---

\* The authors chose to abandon the traditional alphabetical ordering of authors in favor of a randomized ordering, marked by (R). The publicly verifiable documentation of this randomization can be found at: <https://tinyurl.com/2mb96dea>




© Mridul Nandi, N. V. Vinodchandran, Arijit Ghosh, Kuldeep S. Meel, Soumit Pal, and Sourav Chakraborty; licensed under Creative Commons License CC-BY 4.0


Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).


Editors: Amit Kumar and Noga Ron-Zewi; Article No. 26; pp. 26:1–26:20



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

**Funding** *N. V. Vinodchandran* : Vinodchandran’s research is partially supported by NSF grants 2130608 and 2342244.

*Arijit Ghosh* : Arijit Ghosh is partially supported by the Science & Engineering Research Board of the DST, India, through the MATRICS grant MTR/2023/001527.

*Kuldeep S. Meel* : This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019- 0004].

**Acknowledgements** We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2024-05956].

## 1 Introduction

The past three decades bear witness to significant developments in the field of data streaming. The widespread adoption of computing systems has led to the era of big data, wherein the ubiquity of sensors has allowed the collection of a large amount of data. Consequently, the data streaming model and the design of algorithms that balance time and space efficiency in this model are of significant interest to theoreticians and practitioners alike.

In this paper, we focus on one of the fundamental problems in data streaming: Given a stream of sets  $S_1, S_2, \dots, S_M$  where each  $S_i$  is a subset of a universe  $\Omega$ , output an  $(\varepsilon, \delta)$ -estimate (see the Definition 5) of the size of the union of the sets, that is,  $|\bigcup_i S_i|$ . Note that when sets are singletons, the problem boils down to the estimation of the zeroth frequency moment ( $F_0$ ) of the stream of items and is well-studied in streaming literature. In particular, a long line of work culminated in the development of algorithms for  $F_0$ -estimation with optimal space complexity  $O(\log |\Omega| + \frac{1}{\varepsilon^2})$  and  $O(1)$  update time complexity [16] (for a constant error probability  $\delta$ ). In this work, we will focus on the Delphic family of sets which is a general framework for computational problems over abstract sets.

► **Definition 1** (Delphic family). *Let  $\Omega$  be a discrete universe. A set  $S \subseteq \Omega$  belongs to a Delphic family<sup>1</sup> if the following queries can be done in  $O(\log |\Omega|)$  time: (1) Membership: Given any  $x \in \Omega$  check if  $x \in S$ , (2) Cardinality: Determine the size of  $S$ , that is  $|S|$ , (3) Sampling: Draw a uniform random sample from  $S$ .*

The notion of the Delphic family is general enough to capture several well-known problems, such as Klee’s Measure Problem (KMP) [23, 25], test coverage estimation, and DNF counting. For example, the streaming version of Klee’s Measure Problem (KMP) refers to the case where the sets in a stream are represented by an axis-parallel rectangle in  $[\Delta]^d$  where  $\Delta$  is a natural number. KMP is a naturally occurring and fundamental topic that has been extensively researched in computational geometry [4, 6, 8, 10, 9, 12, 14, 17, 21]. While *Delphic Sets* was coined in [19], the notion has been implicit in prior work stretching to the early 1980’s. We are interested in designing streaming algorithms for estimating the size of the union of sets over a Delphic family. We also call this problem  $F_0$  estimation problem over Delphic sets.

► **Problem 2.** *Given a stream  $\mathcal{S} = \langle S_1, S_2, \dots, S_M \rangle$  wherein each  $S_i \subseteq \Omega$  belongs to a Delphic family, and  $0 < \varepsilon < 1$  output an  $(\varepsilon, 1/3)^2$  approximation of  $F_0(\mathcal{S}) := \left| \bigcup_{i=1}^M S_i \right|$ .*

<sup>1</sup> As observed in [18], every Delphic set can be represented by a circuit of size  $O(\log |\Omega| \log \log |\Omega|)$ .

<sup>2</sup>  $c$  is  $(\varepsilon, 1/3)$  approximation of  $F_0(\mathcal{S})$  if  $\Pr[F_0(\mathcal{S})(1 - \varepsilon) \leq c \leq F_0(\mathcal{S})(1 + \varepsilon)] \geq 2/3$ . Note that the choice of  $1/3$  is arbitrary. The probability  $1/3$  can be boosted to an arbitrary  $\delta$  by using standard boosting technique.

■ **Table 1** In the table  $n := \log |\Omega|$ ,  $m := \log M$  and  $c := O(1)$ . The Big-Oh notation ( $O$ ) in the above comparison table shows only dependency on  $n$  and  $m$  and hides the  $\text{poly}(1/\varepsilon, 1/\delta)$  factors on  $\varepsilon$  and  $\delta$ .

| Comparison of Complexity Bounds with Previous Works |                |                |   |   |
|---|----------------|----------------|---|---|
| Algorithm   | Technique Type | Set Type       | Space Complexity                        | Update Time                                       |
| [22]  | Hashing based  | KMP $d = 1$    | $O(\varepsilon^{-2} \cdot n)$           | $O(n)$  |
| [25]  | Hashing based  | KMP $d \geq 2$ | $O(\varepsilon^{-1} \cdot d \cdot n)^c$ | $O(n^d)$  |
| [19]  | Sampling based | Delphic        | $O(\varepsilon^2 \cdot m \cdot n)$      | $O(\varepsilon^2 \cdot \log m \cdot m^2 \cdot n)$ |
| [18]  | Sampling based | Delphic        | $O(\varepsilon^{-2} \cdot n^3)$         | $O(\varepsilon^{-2} \cdot n^4)$                   |
| <b>This Paper</b>                                   | Sampling based | Delphic        | $O(\varepsilon^{-2} \cdot n^2)$         | $O(\varepsilon^{-2} \cdot n^2)$                   |

The design goal is to optimize the algorithm’s update time and space complexity, wherein the update time complexity refers to the amount of time spent processing an item (a set in our case) of the stream.

## 1.1 Prior Work and Technical Challenges

Since the work of Alon, Matias, and Szegedy [1], streaming algorithms gained considerable interest from algorithm design community. However, the problem of estimating distinct elements in a stream of items has been investigated prior to the work of [1]. In particular, the seminal work of Flajolet and Martin [11] pioneered a sketching-based framework for streaming algorithms for the case wherein every element of the stream is a singleton. The sketching-based techniques crucially rely on the use of pairwise independent hash functions. In the early 2000s, the sketching-based approach emerged as a principal technical tool in the design of streaming algorithms. Particularly for  $F_0$  estimation of single-item streams, a series of hash-function-based algorithms led to the development of both space-efficient and update-time optimal algorithms [16, 5].

$F_0$  estimation over set streams has garnered interest from researchers due to the natural extension from singletons to sets. Notably, Pavan and Tirthapura [22] and Sun and Poon [24] explored range-efficient  $F_0$  estimation, which addresses a specific case of the KMP in one dimension. For the broader KMP, Tirthapura and Woodruff [25] developed an algorithm with optimal space complexity. However, the update time for their algorithm was  $O(|\Omega|)$ , which is exponential in terms of set representation. Subsequently, Pavan, Vinodchandran, Bhattacharyya, and Meel [23] proposed an alternative technique, yet it similarly faced an update time complexity of  $O(|\Omega|)$ .

All the above-mentioned algorithms employed hash function based approaches and failed to yield an algorithm with update time complexity polynomial in representation of sets for the general case of Delphic sets. The primary technical barrier to such approaches arises from the fact that sketching-based techniques crucially rely on checking whether for a function  $h$ , randomly chosen from a pairwise independent hash family, there exists an element  $x \in S_i$  such that  $h(x) = 0$ . Nevertheless, whether a pairwise independent hash family exists that supports such checking in time  $\text{poly}(\log |\Omega|)$  is unknown.

In [19], the authors introduced a sampling-based technique for  $F_0$  estimation that eschews traditional hash functions. Their method involves maintaining a bucket  $\mathcal{X}$ , where each stream element is selected independently with probability  $p$ . To handle element repetitions, new elements from a set  $S_i$  replace those in  $\mathcal{X}$ , with all elements in  $S_i$  sampled independently

at the same probability  $p$ . To manage the bucket's size, elements are discarded with a probability of  $1/2$  once a threshold based on  $\varepsilon^{-1}$  is reached, halving  $p$  simultaneously. This approach yields a space complexity of  $\tilde{O}\left(\frac{\log |\Omega| \cdot \log M}{\varepsilon^2}\right)$ , with a logarithmic dependence on the stream size  $M$ . In a subsequent work [18], the authors modified this approach by allowing  $p$  to vary, not just decrease. Each tuple in  $\mathcal{X}$  then included the element and its sampling probability at arrival. This adjustment removed the dependency on  $M$ , leading to space complexity of  $\tilde{O}\left(\frac{\log^3 |\Omega|}{\varepsilon^2}\right)$  and update time complexity of  $\tilde{O}\left(\frac{\log^4 |\Omega|}{\varepsilon^2}\right)$ . This contrasts with the optimal algorithm for singleton sets  $S_i$  by Kane, Nelson, and Woodruff, which has a space complexity of  $O(\log |\Omega| + \frac{1}{\varepsilon^2})$ .

The above-mentioned line of research leads to the following significant open question:

*Can we design algorithm for  $F_0$  estimation over Delphic sets with space complexity  $O(\log |\Omega|)$  and update-time complexity  $\text{poly}(\log |\Omega|)$  (ignoring dependency on  $\varepsilon$  and  $\delta$ )?*

*Remark:* We note that if we have no restriction on the update time complexity, then the problem over Delphic sets reduces to  $F_0$  estimation of singleton streams: when a set  $S_i$  arrives, we can cycle through all elements in the universe  $\Omega$  and use membership testing only to stream elements of  $S_i$ . This leads to an algorithm with optimal space complexity of  $O(\log(|\Omega|) + \varepsilon^{-2})$  but with update time that depends linearly on  $|\Omega|$ . An asymptotic lower bound of  $\Omega(\log(|\Omega|) + \varepsilon^{-2})$  is known for the space complexity [15, 1].

It is worth observing that if we were to follow the approach suggested in [19, 18], then ensuring that the value of  $p$  at least  $\frac{1}{\varepsilon^2 \cdot F_0(S)}$  with sufficiently high probability does entail the space complexity of  $\tilde{O}\left(\frac{\log^3 |\Omega|}{\varepsilon^2}\right)$ . Therefore, an improvement in space complexity must require a new approach.

## 1.2 Our Results

As our first contribution, we report progress towards the above question. In particular, we establish the following:

► **Theorem 3 (Main Theorem).** *There is a streaming algorithm that given a stream  $S = \langle S_1, S_2, \dots, S_M \rangle$  wherein each  $S_i$  belongs to Delphic family, and  $0 < \varepsilon < 1$  outputs an  $(\varepsilon, 1/3)$ -estimation of  $\left| \bigcup_{i=1}^M S_i \right|$  with space complexity  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right)$  and update time complexity  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log^3 \frac{1}{\varepsilon}\right)$ .*

The above theorem improves the space complexity by a factor of  $\log |\Omega|$  and the update time complexity by a factor of  $\log^2 |\Omega|$ . We note that for special cases such as KMP, we can bring the update time complexity factor of  $\log^3 \frac{1}{\varepsilon}$  to  $\log \frac{1}{\varepsilon}$ , resulting in update time complexity of  $O\left(\frac{d^2 \log^2 \Delta}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right)$  where  $\Omega = [\Delta]^d$ .

Theorem 3 leads to the natural question of whether further improvement is possible towards solving the open question of designing a streaming algorithm for  $F_0$  estimation over Delphic sets with space complexity  $O(\log |\Omega|)$  (ignoring the dependence on  $\varepsilon$  and  $\delta$ ) and update time complexity significantly smaller than  $|\Omega|$ . This seems hard and will require new techniques: since storing a single element takes  $O(\log |\Omega|)$  space implies that one can only store a constant number of elements at any point of time. This appears to be a major technical restriction for sampling-based approaches like our algorithm in Theorem 3 and we even conjecture that  $\omega(\log |\Omega|)$  space is required if we restrict the update time to be  $\text{poly}(\log |\Omega|)$ . Unfortunately, establishing such a lower bound has a major computational



complexity bottleneck. In particular, we show that establishing a lower bound on space other than the lower bound known for computing  $F_0$  for singleton streams will lead to major separation result in computational complexity.

$\text{NTISP}(\text{poly}, \text{Linspace})$  is the class of language accepted by non-deterministic Turing machines in polynomial time and linear space simultaneously. Likely,  $\text{DTISP}(\text{poly}, \text{Linspace})$  is the class of languages accepted by deterministic Turing machines in polynomial time and linear space simultaneously. Whether or not  $\text{NTISP}(\text{poly}, \text{Linspace}) = \text{DTISP}(\text{poly}, \text{Linspace})$  is an open problem in complexity theory. If  $\text{NTISP}(\text{poly}, \text{Linspace}) = \text{DTISP}(\text{poly}, \text{Linspace})$  then  $P=NP$ . The other implication is yet to be discovered. However, separating  $\text{NTISP}(\text{poly}, \text{Linspace})$  from  $\text{DTISP}(\text{poly}, \text{Linspace})$  is a hard open question. The best-known lower bound for time-space complexity classes is that SAT (which is in  $\text{NTISP}(\text{poly}, \text{Linspace})$ ) cannot be solved in  $\text{DTISP}(n^{1.8}, o(n))$  [26]. Any improvement on this will be a major result in complexity theory. We show the following:

► **Theorem 4.** *There is a streaming algorithm,  $\text{DelphicWithNP}$ , that given, a stream  $\mathcal{S} = \langle S_1, \dots, S_M \rangle$ , where each set  $S_i \subseteq \Omega$  is a member of a Delphic family,  $0 < \varepsilon$ , and an oracle access to a language belonging to  $\text{NTISP}(\text{poly}, \text{Linspace})$ , computes a  $(\varepsilon, 1/3)$ -approximation of  $F_0(\mathcal{S})$ . Moreover,  $\text{DelphicWithNP}$  has the following properties:*

- (1) *it takes  $O(\log |\Omega| \cdot \varepsilon^{-2})$  space and  $\text{poly}(\log |\Omega|, \frac{1}{\varepsilon})$  update time,*
- (2) *the queries it makes to the oracle are of size  $O(\log |\Omega| \cdot \varepsilon^{-2})$ .*

*Thus, if  $\text{NTISP}(\text{poly}, \text{Linspace}) = \text{DTISP}(\text{poly}, \text{Linspace})$ , there is an algorithm (without oracle calls) for  $(\varepsilon, 1/3)$ -estimation of  $F_0$  of a set stream over Delphic family with  $O(\log |\Omega| \cdot \varepsilon^{-2})$  space and  $\text{poly}(\log |\Omega|, 1/\varepsilon)$  update time.*

Thus, for constant  $\varepsilon$ , we get a space optimal algorithm for estimating  $F_0$  over Delphic set streams as  $\log |\Omega|$  space is required even in the singleton case. In fact, the above theorem is true for a very relaxed version of Delphic sets where we only require one of the conditions for membership in Delphic sets to be decided in time linear in the representation of the set. Theorem 4 implies that establishing a space lower bound of  $\omega(\log |\Omega|)$  for estimating  $F_0(\mathcal{S})$  of Delphic set streams with  $\text{poly}(\log |\Omega|)$  will lead to proving  $\text{NTISP}(\text{poly}, \text{Linspace}) \neq \text{DTISP}(\text{poly}, \text{Linspace})$ , thus resolving a significant lower bound in complexity theory.

Our work leaves a tantalizing open question:

*What is the optimal space complexity for  $F_0$  estimation of set streams over Delphic family with  $\text{poly}(\log |\Omega|)$  update time complexity?*

### 1.3 Technical Overview

#### Key Ideas for Theorem 3

The high-level idea is to maintain a bucket for every level  $k \in \{1, \dots, \log |\Omega|\}$  such that the bucket,  $\mathcal{X}^{(k)}$ , at level  $k$  consists of elements from  $\bigcup_i S_i$  that were independently selected with a probability of  $2^{-k}$ . To handle repetitions, i.e., when a new set  $S_i$  arrives, we remove all the elements of  $\mathcal{X}^{(k)}$  that are present in  $S_i$ . This process ensures that the event of an element  $s$  being in  $\mathcal{X}^{(k)}$  at the end of the stream depends only on whether  $s$  was independently chosen from the set  $S_i$ , where  $S_i$  is the last set containing  $s$ , and there is no  $S_j$  with  $j > i$  such that  $s \in S_j$ .

To bound the space complexity, we establish a threshold, denoted by  $\text{thresh}$ , limiting the maximum size of the bucket. Whenever  $|\mathcal{X}^{(k)}| = \text{thresh}$ , no additional elements can be added to  $\mathcal{X}^{(k)}$  (i.e., we must wait until some elements are removed from  $\mathcal{X}^{(k)}$ ). We set  $\text{thresh} := O\left(\frac{1}{\varepsilon^2}\right)$ . Note that storing any element requires  $O(\log |\Omega|)$  space, and thus storing  $\text{thresh} \log |\Omega|$  elements necessitates  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2}\right)$  space.

The key technical insight from [19] is that if  $\text{thresh} = O\left(\frac{\log M}{\varepsilon^2}\right)$ , it can be demonstrated that, with sufficiently high probability and for  $k > \log(F_0(\mathcal{S}))$ ,  $\mathcal{X}^{(k)}$  would not be full (i.e.,  $|\mathcal{X}^{(k)}| < \text{thresh}$ ) at all times. Furthermore, in [18], it was observed that  $\text{thresh} = O\left(\frac{\log |\Omega|}{\varepsilon^2}\right)$ , then it can be demonstrated that, with sufficiently high probability and for  $k > \log(F_0(\mathcal{S}))$ ,  $\mathcal{X}^{(k)}$  would not be full (i.e.,  $|\mathcal{X}^{(k)}| < \text{thresh}$ ) when an element  $s \in \bigcup_i S_i$  appears for the last time in a stream.

The major technical advancement in our analysis hinges on a crucial observation: while it is plausible for  $\mathcal{X}^{(k)}$  to be full when an element  $s$  makes its final appearance in the stream, the resulting relative error of our estimator  $|\mathcal{X}^{(k)}| \cdot 2^k$  remains manageable even when  $k$  is approximately logarithmic in the order of magnitude of the size of the stream, that is  $k \sim \log(F_0(\mathcal{S}))$ . The ensuing technical analysis is complex due to its dependence on a meticulous formulation of significant events and the construction of a sum of products of random variables. We introduce two sets of random variables, denoted as  $X_{i,r}^{(k)}$  and  $Y_{i,r}^{(k)}$ . The random variable  $X_{i,r}^{(k)}$  indicates whether an element is sampled from the  $j$ -th set  $S_j$  in the stream. The random variables  $Y_{i,r}^{(k)}$  indicate whether a set of sampled elements is included in the buckets. We then define a series of random variables  $Z_r^{(k)}$ , dependent on the aforementioned random variables  $X_{i,r}^{(k)}$  and  $Y_{i,r}^{(k)}$ . To leverage concentration inequalities effectively, we maintain a collection of  $O(1/\varepsilon^2)$  buckets at each level  $k$ . Additionally, we set the size of each bucket to be  $O(\log(1/\varepsilon))$ . At the end of the stream, for each level  $k$ , we calculate the average number of elements in the buckets at that level, denoted as  $\bar{Z}^{(k)}$ .

Finally, it is important to note that our technical analysis only ensures that the relative error is small for  $k \approx \log(F_0(\mathcal{S}))$ . Since  $F_0(\mathcal{S})$  is unknown a priori, we must identify a method to choose an optimal  $k^*$  for returning the final estimate  $\bar{Z}^{(k^*)} \cdot 2^{k^*}$ . A key observation is that  $\bar{Z}^{(k)}$  is less than 1 for  $k \gg \log(F_0(\mathcal{S}))$ . Therefore, finding the largest  $k$  for which  $\bar{Z}^{(k)}$  exceeds 1 should suffice.

## Key Ideas for Theorem 4

The main idea is to adapt one of the standard hashing-based algorithms (e.g., Gibbons and Tirthapura's algorithm) for  $F_0$  estimation for singleton streams that takes  $O(\log |\Omega| \cdot \frac{1}{\varepsilon^2})$  space. The algorithm starts with picking a hash function  $h$  from a pairwise independent family and keeps a bucket  $\mathcal{X}$  which is initially set to empty and has a capacity of  $O(\frac{1}{\varepsilon^2})$ . The computational challenge is to implement the update step. For this, when a new set  $S$  comes, we need to add all elements  $x \in S$  so that the first  $m$  bits of  $h(x)$  are all 0s for a variable  $m$  that the algorithm keeps. If the addition of these elements makes the bucket  $\mathcal{X}$  overflow,  $m$  is incremented and deletes all elements  $\mathcal{X}$  with  $m + 1$ st bit of hash value is non-zero. The main computational challenge is to implement this step. However, we show how this step can be implemented by making linear size queries to disjoint union of two languages in NTISP(poly,LINSPACE). Thus if NTISP(poly,LINSPACE) = DTISP(poly,LINSPACE), then queries to this language can be simulated in space  $O(\log |\Omega| \cdot \frac{1}{\varepsilon^2})$  and update time  $\text{poly}(\log |\Omega|, \varepsilon^{-1})$ .

## 1.4 Paper Organization

The remainder of this paper is structured as follows. Section 2 introduces key notations and fundamental concepts. In Section 3, we present our primary Algorithm 2 for Delphic sets with WOR sampling, along with its correctness analysis. Subsequently, in subsection 3.2, we

present the algorithm for the general Delphic set. Section 4 contains the proof of Theorem 4. The appendix is divided as follows: Section A presents the concentration bounds used in the correctness analysis; Section B provides the missing proofs for Proposition 15; Section C details the algorithm for KMP along with its correctness analysis; and finally, Section D offers some basic probability results.

## 2 Preliminaries

### 2.1 $F_0$ -Estimator and Klee's Measure Problem

For a stream of sets,  $\mathcal{S} := \langle S_1, S_2, \dots, S_M \rangle$  where each set in the stream is a subset of  $\Omega$ , recall that we denote by  $F_0(\mathcal{S})$  the size of the union of the sets:  $F_0(\mathcal{S}) = \left| \bigcup_{j=1}^M S_j \right|$

► **Definition 5.** *A random variable  $X$  is an  $(\varepsilon, \delta)$ -approximation of  $c$  if  $\Pr[(1 - \varepsilon)c \leq X \leq (1 + \varepsilon)c] \geq (1 - \delta)$ . We also simply write the event  $(1 - \varepsilon)c \leq X \leq (1 + \varepsilon)c$  as  $X = (1 \pm \varepsilon)c$  (or  $X \neq (1 \pm \varepsilon)c$  to denote the complement event).*

Finally an  $F_0$ -estimator is a streaming algorithm that on input  $\varepsilon, \delta \in (0, 1)$  and access to a stream of set  $\mathcal{S} := \langle S_1, S_2, \dots, S_M \rangle$  outputs an  $(\varepsilon, \delta)$  approximation of  $F_0(\mathcal{S})$ .

There are two main complexity measures that we are interested about an  $F_0$ -estimator - space complexity and update time complexity. The space complexity is the amount of work space needed by the algorithm. The update time complexity is the amount of time spent by the algorithm for processing a single set in the stream. The goal is to design an  $F_0$ -estimator while trying to minimize both the space complexity and the update time complexity of the algorithm.

The notion of Delphic sets captures several well-known problems, such as Klee's Measure Problem, which we define below.

Let  $\Delta$  be a natural number, consider the following set  $[\Delta] = \{1, 2, \dots, \Delta\}$ . A  $d$ -dimensional axis-aligned rectangle  $\mathbf{r}$  over  $[\Delta]^d$  is a subset of  $[\Delta]^d$ , succinctly represented by the tuple  $(a_1, b_1, \dots, a_d, b_d)$ , and contains all the tuples  $\{(x_1, \dots, x_d)\}$  where  $a_i \leq x_i \leq b_i$  and  $x_i \in [\Delta]$ . Formally, we write  $\mathbf{r} = \{(x_1, x_2, \dots, x_d) : a_i \leq x_i \leq b_i, x_i \in [\Delta] \text{ for all } i \in [d]\}$

► **Definition 6** (Klee's Measure Problem (KMP) in Streaming Setting). *Given a stream  $\mathcal{R}$  of size  $M$  such that  $\mathcal{R} = \langle \mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_M \rangle$ , where each item  $\mathbf{r}_i$  is a  $d$ -dimensional rectangle, compute a  $(\varepsilon, \delta)$ -approximation of  $\left| \bigcup_{i=1}^M \mathbf{r}_i \right|$ .*

Note that KMP is a special case of Problem 2 since set of  $d$ -dimensional axis-aligned rectangles over  $[\Delta]^d$  forms a Delphic family.

### 2.2 Notations

For any positive integer  $k$ , we write  $[k] = \{1, 2, \dots, k\}$ . In the rest of this paper, we will assume that  $\mathcal{S} := \langle S_1, S_2, \dots, S_M \rangle$  is a stream of sets, where each set is a subset of universe  $\Omega$ , i.e.,  $S_j \subseteq \Omega$  for all  $j \in [M]$ . We denote by  $s_j := |S_1| + \dots + |S_j|$  for all  $j \in [M]$  and  $s_0 := 0$ . We write  $s := s_M$  to denote the total number of elements counting with repetition that appeared in the stream  $\mathcal{S}$ . For  $j \in [M]$ , let  $\mathcal{I}_j = \{s_{j-1} + 1, s_{j-1} + 2, \dots, s_j\}$ , the set  $\mathcal{I}_j$  denotes the indices of elements in the set  $S_j$ . Note that  $\mathcal{I}_j$ 's are disjoint and  $\bigcup_{j=1}^M \mathcal{I}_j = [s]$ . Clearly, for every  $i \in [s]$ , we can assign a *streaming-index*  $j := j(i)$  (which is unique) such that  $i \in \mathcal{I}_j$ .

Although, total order of the elements on  $\Omega$  is not needed for our algorithms, we will assume an ordering of the elements of  $\Omega$ ; this will help us in the presentation of the correctness proofs of the algorithm. We can open the set stream into element streams and write a sequence  $\langle x_1, \dots, x_s \rangle$ , where  $S_j = \{x_i : i \in \mathcal{I}_j\}$ ,  $x_{s_{j-1}+1} < \dots < x_{s_j} \forall j \in [M]$ . For any  $k \in [|S_j|]$  we will denote by  $S_j[k]$  to the element  $x_{s_{j-1}+k}$ .

At any point, say after the sets  $S_1, \dots, S_j$  have arrived in the stream and for any element  $x$  in the stream  $\langle x_1, \dots, x_{s_j} \rangle$ , that is, the stream we have seen till now, we will be interested in the last time the element appeared in the stream. The set of indices in  $[s_{j-1}]$  that contains the last appearance of any element is called the set of *final indices with respect to  $j$ th set  $S_j$*  and is denoted by  $\mathcal{F}_j$ .

More formally, let  $j \in [M]$  be a stream index. We call  $i \leq s_{j-1}$  a *final index with respect to  $j$ th set  $S_j$*  if  $x_i \notin \{x_{i+1}, \dots, x_{s_j}\}$ . Let

$$\mathcal{F}_j = \{i \leq s_{j-1} : i \text{ is a final indices w.r.t. } j\text{th set } S_j\}.$$

When  $j = 1$ , the set  $\mathcal{F}_1 = \emptyset$ .

Now we make the following simple but valuable observations:

► **Observation 7.**

1. For all  $j \in [M]$ , (i)  $\mathcal{F}_j$  and  $\mathcal{I}_j$  are disjoint i.e.,  $\mathcal{F}_j \cap \mathcal{I}_j = \emptyset$  and (ii)  $\{x_i : i \in \mathcal{F}_j\}$  is disjoint from  $S_j$ .
2.  $\{x_i : i \in \mathcal{F}_j\} \sqcup S_j \subseteq S$ . The equality occurs for  $j = M$ , i.e.,  $\{x_i : i \in \mathcal{F}\} = S$  where  $\mathcal{F} = \mathcal{F}_M \sqcup \mathcal{I}_M$ . Note,  $|\mathcal{F}| = F_0(S)$ .
3. For all  $j \in [M - 1]$ ,  $\mathcal{F}_{j+1} \subseteq \mathcal{F}_j \sqcup \mathcal{I}_j$ .

## 2.3 Delphic Family with WOR Sampling

The definition of Delphic Sets (Definition 1) allows one to draw a uniform random sample from a Delphic Set  $S$ . However, if one needs to uniformly draw a set of  $k$  distinct samples from  $S$ , the only option is to draw independent samples from  $S$  until one gets  $k$  distinct samples. One can use the coupon collector theorem to ensure that with high probability, one has to draw at most  $O(k \log k)$  samples. Nevertheless, this “high probability” statement may not be sufficient if one has to repeat this process multiple times - a slightly more involved calculation may be necessary.

One way of dealing with this issue is assuming one is allowed to draw samples “without replacement (WOR)”. Formally, we define a Delphic family with WOR sampling as follows:

► **Definition 8** (Delphic family with WOR sampling). *A set  $S \subseteq \Omega$  belongs to the Delphic family with WOR sampling if the following queries can be done:*

- (1) know the size of the set  $S$ , in  $O(\log |\Omega|)$  time
- (2) given any  $x$  check if  $x \in S$ , in  $O(\log |\Omega|)$  time
- (3) for any  $k \leq |S|$ , we can draw in  $O(k \log |\Omega|)$  time a uniformly random subset  $S'$  of size  $k$ .

Note that, in contrast to Definition 8, the original Delphic family (Definition 1) can be called “Delphic family with WR sampling”, where WR stands for “with replacement”.

## 2.4 Random Processes and Distributions

We will be drawing samples according to different distributions. All the distributions are standard in the literature. The following are the distributions we will be using in this paper.

► **Definition 9** (Binomial Distribution). *Given any  $n \in \mathbb{N}$  and any  $p \in [0, 1]$  the Binomial Distribution over the set of integers  $\{0, 1, 2, \dots, n\}$  is denoted as  $\text{Bin}(n, p)$  where probability of a number  $0 \leq k \leq n$  is  $\binom{n}{k} p^k (1 - p)^{n-k}$ .*

► **Definition 10** (Bernoulli Process). *Given any finite set  $S$  and any  $p \in [0, 1]$ , a Bernoulli sample of probability  $p$  for the set  $S$  is denoted as  $\text{Ber}(S, p)$  such that every element  $x \in S$  is picked with probability  $p$ . If  $X$  denotes the sampled subset of  $S$  with probability  $p$  then for any set  $T \subseteq S$ ,  $\Pr(X = T) = p^{|T|} (1 - p)^{|S| - |T|}$ .*

*By abuse of notation, we will use  $\text{Ber}(p)$  to denote the distribution over  $\{0, 1\}$  where the probability of 1 is  $p$ . So, the Bernoulli process is nothing but independent copies of the Bernoulli distribution where 1 (or 0) represents that the element is picked (or not picked respectively).*

Clearly, if  $X \sim \text{Ber}(S, p)$  then  $|X| \sim \text{Bin}(|S|, p)$ . We now describe how to sample the Bernoulli process. We first sample  $d \sim \text{Bin}(|S|, p)$ , then we sample  $d$  many elements, denoted as  $T$ , in a without replacement (WOR) manner. Then,  $T \sim \text{Ber}(S, p)$ . One can sample WOR through with replacement (WR) sampling (this is well-known as the coupon collection problem). We have the following lemma using the coupon collector theorem (see Appendix D).

► **Lemma 11.**  $\text{WRSample}(S, r, t)$  (Algorithm 1) outputs a subset  $L$  of the set  $S$  with the following guarantee

- With probability  $\geq \left(1 - r^{-\frac{t}{r \log r} + 1}\right)$  the algorithm outputs a set of size  $r$  distinct samples, each drawn uniformly from the set  $S$ .
- With the remaining probability at most  $r^{-\frac{t}{r \log r} + 1}$  the algorithm outputs an empty set.
- The maximum number of samples drawn from  $S$  is  $t$ .

■ **Algorithm 1**  $\text{WRSample}(S, r, t)$ .

---

```

1: Input Set  $S$   $t, r \in \mathbb{N}$ 
2: Initialize  $L = \emptyset$ ;
3: for  $1 \leq i \leq t$  do
4:   if  $|L| < r$  then
5:     Draw a random sample  $x$  from  $S$  (with replacement)
6:     if  $x \notin L$  then  $L = L \cup \{x\}$ 
7: if  $|L| \neq r$  then  $L = \emptyset$ 
8: Output  $L$ 

```

---

**3 F<sub>0</sub>-Estimator for Delphic sets**

In this section, we prove Theorem 3. We prove it in two steps. In Section 3.1, we present Algorithm 2 and prove that it is an  $F_0$ -estimator for Delphic Sets with WOR sampling. Then, in Section 3.2, we show how the Algorithm 2 can be modified to obtain an  $F_0$ -estimator for general Delphic Sets and hence prove Theorem 3.

**3.1 Handling Delphic Sets with WOR sampling**

In this section, we will prove the following theorem:

► **Theorem 12.** *If  $\mathcal{S}$  is a stream of Delphic Sets with WOR sampling, then the Algorithm 2 is an  $(\varepsilon, 1/3)$ - $F_0$ -estimator. Also, the space and update time complexity of Algorithm 2 is  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$ .*

Before we present the proof of the correctness of the algorithm and the analysis of its complexities (in Section 3.1.1 and Section 3.1.2 respectively), we will give a brief description of the algorithm.

### Description of the Algorithm

Let  $0 < \varepsilon \leq 1/2$ , and we set  $\text{thresh} := \max\{18, \lceil \log \frac{2}{\varepsilon} \rceil\}$  and  $\text{Reps} = 90/\varepsilon^2$ . We will now give an efficient  $F_0$ -estimator for Delphic sets. At each level  $k$ , where  $k \in \{1, \dots, \log |\Omega|\}$ , there is a collection of  $\text{Reps}$  many buckets  $\mathcal{X}_r^{(k)}$  with  $r \in \{1, \dots, \text{Reps}\}$ . Each bucket  $\mathcal{X}_r^{(k)}$  can contain at most  $\text{thresh}$  many samples from the universe  $[n]$ . At the beginning of the algorithm, all the  $\text{Reps}$  many buckets in each level  $k$  is an empty set. Let the input stream  $S$  of Delphic sets be the following:  $\mathcal{S} := \langle S_1, \dots, S_M \rangle$ , where  $S_j \subseteq \Omega$  for all  $j \in [M]$ . Note that we want to estimate the size of the set  $S = \bigcup_{j=1}^M S_j$ . Suppose that our algorithm has already processed

the  $S_1, \dots, S_i$  from the stream and the buckets  $\mathcal{X}_r^{(k)}$  in each level  $k$  contains a sample of elements from the set  $S$ . Now, our algorithm receives a new set  $S_{i+1}$  over the data stream. Our algorithm will perform the following steps to process the new set  $S_{i+1}$ :

**Step 1:** For each buckets  $\mathcal{X}_r^{(k)}$ , where  $k \in [\log |\Omega|]$  and  $r \in [\text{Reps}]$ , we will first begin by removing all the elements in  $\mathcal{X}_r^{(k)}$  that are also present in the new set  $S_{i+1}$ , see **lines** 6 to 9 from the Algorithm 2. Observe that the time complexity for this step will be  $O(\text{Reps thresh} \log^2 |\Omega|)$

**Step 2:** Ideally, we would like to add an element from  $S_{i+1}$  independently with probability  $2^{-k}$  to each bucket  $\mathcal{X}_r^{(k)}$ , but this may not be possible as the size of each bucket is  $\text{thresh}$ . To work around this problem, for each  $k \in [\log |\Omega|]$  and  $r \in [\text{Reps}]$ , we will first sample  $d_r^{(k)} \sim \text{Bin}(|S_{i+1}|, 2^{-k})$ . By slight abuse of notation, we will denote by  $|\mathcal{X}_r^{(k)}|$  the current size of the bucket  $\mathcal{X}_r^{(k)}$ . If  $d_r^{(k)} \geq \text{thresh} - |\mathcal{X}_r^{(k)}|$  then we will keep  $\mathcal{X}_r^{(k)}$  unchanged. Otherwise, as  $S_{i+1}$  is a Delphic set with WOR sampling, we will sample without replacement  $d_r^{(k)}$  many samples from  $S_{i+1}$  and add them to  $\mathcal{X}_r^{(k)}$ . See **lines** 10 to 13 from the Algorithm 2.

Once we have processed the whole stream  $\mathcal{S} = \langle S_1, \dots, S_M \rangle$ , for all  $k \in [\log |\Omega|]$ , we will calculate the average size  $\bar{Z}^{(k)}$  of the buckets in each level  $k$ , (See **lines** 14 to 15 from Algorithm 2) that is,

$$\bar{Z}^{(k)} = \frac{1}{\text{Reps}} \sum_{r=1}^{\text{Reps}} |\mathcal{X}_r^{(k)}|$$

Once all the averages have been computed for each level  $k \in [\log |\Omega|]$  the algorithm computes the maximum  $k$ , let us call it  $k^*$  for which  $\bar{Z}^{(k^*)} \geq 1$ . Finally, the algorithm outputs  $2^{k^*} \cdot \bar{Z}^{(k^*)}$ . See **lines** 16 to 17 from Algorithm 2.

### 3.1.1 Correctness of Algorithm 2 for Delphic Sets with WOR sampling

If we replace the **lines** 10-13 in Algorithm 2 by the following lines, then it is easy to see that it functionally behaves identically.

10: Draw  $d_r^{(k)}$  from  $\text{Bin}(|S_j|, 1/2^k)$

■ **Algorithm 2** [ $F_0$ -Estimator for Delphic Sets with WOR sampling.]

---

```

1: Input Stream =  $\langle S_1, S_2, \dots, S_M \rangle$ ,  $\varepsilon$ ,  $\delta$ 
2: Initialize
3:  $p \leftarrow 1$ ; thresh  $\leftarrow \max\{\lceil \log 2/\varepsilon \rceil, 18\}$ ; Reps  $\leftarrow \lceil 90/\varepsilon^2 \rceil$ ;
4: for ( $1 \leq k \leq \log |\Omega|$ ) and ( $1 \leq r \leq \text{Reps}$ ) do
5:    $\mathcal{X}_r^{(k)} \leftarrow \emptyset$ ;

   STREAMING PHASE:
6: for  $j = 1$  to  $M$  do.
7:   for ( $1 \leq k \leq \log |\Omega|$ ) and ( $1 \leq r \leq \text{Reps}$ ). do
8:     for all  $s \in \mathcal{X}_r^{(k)}$  do
9:       if  $s \in S_j$  then remove  $s$  from  $\mathcal{X}_r^{(k)}$ 
10:    Draw  $d_r^{(k)}$  from  $\text{Bin}(|S_j|, 1/2^k)$ 
11:    if  $|\mathcal{X}_r^{(k)}| + d_r^{(k)} \leq \text{thresh}$  then
12:       $\mathcal{L}_r^{(k)}$  is a set of  $d_r^{(k)}$  samples drawn from  $S_j$  WOR
13:       $\mathcal{X}_r^{(k)} = \mathcal{X}_r^{(k)} \cup \mathcal{L}_r^{(k)}$ 

   POST-STREAMING PHASE:
14: for  $1 \leq k \leq \log |\Omega|$  do.
15:    $\bar{Z}^{(k)} = \frac{1}{\text{Reps}} \sum_{r=1}^{\text{Reps}} |\mathcal{X}_r^{(k)}|$ 
16:  $k^* = \max\{k \in [\log n] : \bar{Z}^{(k)} \geq 1\}$ 
17: Output  $\bar{Z}^{(k^*)} \cdot 2^{k^*}$ 

```

---

```

11:  $\mathcal{L}_r^{(k)}$  is a set of  $d_r^{(k)}$  samples drawn  $S_j$  WOR
12: if  $|\mathcal{X}_r^{(k)}| + d_r^{(k)} \leq \text{thresh}$  then
13:    $\mathcal{X}_r^{(k)} = \mathcal{X}_r^{(k)} \cup \mathcal{L}_r^{(k)}$ 

```

Note that the differences between the above lines and the lines in the Algorithm 2 is that in the Algorithm 2 we do not sample the set  $\mathcal{L}_r^{(k)}$  unless we are sure that we will use the set  $\mathcal{L}_r^{(k)}$ , that is the condition  $|\mathcal{X}_r^{(k)}| + d_r^{(k)} \leq \text{thresh}$  is satisfied. We do this in the actual algorithm (more like a “lazy sampling”) to have better control on the worst-case update time complexity. But, for the presentation of the proof of correctness of the theorem it is useful to use the above three lines (instead of **lines** 10-13 in Algorithm 2) where we first sample  $\mathcal{L}_r^{(k)}$  and then decide whether to use it.

Let us consider the  $j$ th round of the streaming phase of the algorithm. That is when the set  $S_j$  arrives in the stream. By the construction of the set  $\mathcal{L}_r^{(k)}$  it follows that all elements of  $S_j$ 's are chosen in  $\mathcal{L}_r^{(k)}$  independently with probability  $2^{-k}$ . For the sake of presentation, let  $\mathcal{L}_{j,r}^{(k)}$  denote the set  $\mathcal{L}_r^{(k)}$  during the  $j$ th round of the streaming phase of the algorithm<sup>3</sup>. Recall that the elements in the set  $S_j$ , that is, the elements  $x_{s_{j-1}+1}, \dots, x_{s_j}$  and for any  $i \in [s_j]$  we denote by  $j(i)$  the stream index such that  $i \in \{s_{j(i)-1} + 1, \dots, s_{j(i)}\}$  (refer to the notations in the Section 2.2). For any  $i \in [s_j]$  let us denote by  $X_{i,r}^{(k)}$  the random variable that takes value 1 if  $x_i$  is included in  $\mathcal{L}_{j(i),r}^{(k)}$  and 0 otherwise, that is, the element  $x_i$  is in the set  $\mathcal{L}_r^{(k)}$  during the  $j$ th round of the streaming phase of the algorithm. Clearly, for any  $i, r, k$   $X_{i,r}^{(k)} \stackrel{i.i.d.}{\sim} \text{Ber}(2^{-k})$ .

---

<sup>3</sup> We ignore the streaming index  $j$  in the Algorithm 2 as we use the same state to update the random set over different indices  $j$ .

However, the random set is actually included in  $\mathcal{X}_r^{(k)}$  provided  $|\mathcal{X}_r^{(k)}| + d_r^{(k)} \leq \text{thresh}$ . To indicate whether an element is actually included in  $\mathcal{X}_r^{(k)}$  we define the random variable  $Y_{i,r}^{(k)}$ , for any  $i \in [s_j]$ . We define the random variable  $Y_{i,r}^{(k)}$  as  $Y_{i,r}^{(k)} = 1$  if  $|\mathcal{X}_r^{(k)}| + d_r^{(k)} \leq \text{thresh}$ , 0 otherwise (i.e.,  $x_i$  is eventually included in  $\mathcal{X}_r^{(k)}$  if  $X_{i,r}^{(k)} Y_{i,r}^{(k)} = 1$ ).

Formally, the random variable  $Y_{i,r}^{(k)}$  can be defined recursively. First we note that since in **lines** 8-9 (of Algorithm 2) we remove all the elements in  $\mathcal{X}_r^{(k)}$  that are in  $S_j$ , so at that time (after **line** 9)

$$\mathcal{X}_r^{(k)} \subseteq \left( \cup_{k=1}^{j-1} S_k \right) \setminus S_j = \{x_i : i \in \mathcal{F}_j\}.$$

Thus the size of  $\mathcal{X}_r^{(k)}$  after **line** 9 is  $\sum_{a \in \mathcal{F}_j} X_{a,r}^{(k)} Y_{a,r}^{(k)}$ . Thus,

$$Y_{i,r}^{(k)} = \begin{cases} 0 & \text{if } \sum_{a \in \mathcal{F}_j(i)} X_{a,r}^{(k)} Y_{a,r}^{(k)} + \sum_{a \in \mathcal{I}_j(i)} X_{a,r}^{(k)} > \text{thresh} \\ 1 & \text{otherwise.} \end{cases}$$

Now, in the post-streaming phase, we set

$$Z_r^{(k)} = \sum_{i \in \mathcal{F}} X_{i,r}^{(k)} Y_{i,r}^{(k)}, \forall r \in [\text{Reps}], \quad \bar{Z}^{(k)} = \frac{Z_1^{(k)} + \dots + Z_{\text{Reps}}^{(k)}}{\text{Reps}}.$$

Note that  $\mathcal{F}$  denote the set of all final indices of the elements of  $S$ . Thus,  $Z_r^{(k)}$ 's are the sizes of the sets  $\mathcal{X}_r^{(k)}$  after processing the stream. Moreover, these are independent for all  $r$  and  $k$ .

► **Lemma 13.** For any  $k$  such that  $\text{thresh} \geq 6c$  where  $c := \frac{F_0(S)}{2^k}$ .

$$c(1 - 2^{-\text{thresh}}) \leq \mathbb{E}(Z_r^{(k)}) \leq c \tag{1}$$

$$\text{Var}(Z_r^{(k)}) \leq c(1 + 2^{-\text{thresh}})c. \tag{2}$$

**Proof.** The upper bound of expectation directly follows from the linearity of expectation. For the lower bound, we first note that

$$\begin{aligned} \Pr(Y_{i,r}^{(k)} = 0 \mid X_{i,r}^{(k)} = 1) &\leq \Pr \left( \sum_{a \in \mathcal{F}_j} X_{a,r}^{(k)} Y_{a,r}^{(k)} + \sum_{a \in \mathcal{I}_j} X_{a,r}^{(k)} \geq \text{thresh} + 1 \mid X_{i,r}^{(k)} = 1 \right) \\ &\leq \Pr(\text{Bin}(F_0(S) - 1, p) \geq \text{thresh}) \leq 2^{-\text{thresh}}. \end{aligned}$$

Note that the last inequality follows from Lemma 20. Thus,  $\mathbb{E}(X_{i,r}^{(k)} Y_{i,r}^{(k)}) = \Pr(X_{i,r}^{(k)} = 1, Y_{i,r}^{(k)} = 1) \geq p(1 - 2^{-\text{thresh}})$  for all  $i$ . Hence, the lower bound is established by linearity of expectation. To bound the variance, we first bound the second moment:

$$\mathbb{E}((Z_r^{(k)})^2) = \mathbb{E}(Z_r^{(k)}) + \sum_{i \neq j} \mathbb{E}(X_{i,r}^{(k)} Y_{i,r}^{(k)} X_{j,r}^{(k)} Y_{j,r}^{(k)}) \leq \mathbb{E}(Z_r^{(k)}) + F_0(S)^2 p^2.$$

Hence we obtain,

$$\begin{aligned} \text{Var}(Z_r^{(k)}) &\leq \mathbb{E}(Z_r^{(k)})(1 - \mathbb{E}(Z_r^{(k)}) + F_0(S)^2 p^2) \\ &\leq c(1 - (1 - 2^{-\text{thresh}})c) + c^2 \leq c + c^2 2^{-\text{thresh}} \end{aligned} \quad \blacktriangleleft$$



Before we complete the proof of the correctness of our Algorithm 2, we first state a core result based on bucket-load random variables (defined below) that would be used.

► **Definition 14** (Bucket-load). *A  $(c, \alpha)$ -bucket-load is a nonnegative random variable  $Z$  such that the first and second moments of  $Z$  satisfy the following relations:*

$$c(1 - \alpha) \leq \text{Ex}(Z) \leq c \quad \text{AND} \quad \text{Var}(Z) \leq c + \alpha c^2.$$

From Lemma 13 we note that for any  $k$  with  $\text{thresh} \geq 6F_0(\mathcal{S})/2^k$ ,  $Z_r^{(k)}$ 's are  $(c, \alpha)$ -bucket-load random variables for  $c = F_0(\mathcal{S})/2^k$  and  $\alpha = 2^{-\text{thresh}} \leq \min\{\varepsilon/2, 2^{-18}\}$ . Due to independence of the random variables  $Z_1^{(k)}, \dots, Z_{\text{Reps}}^{(k)}$ , the random variable  $\bar{Z}^{(k)}$  has mean approximately  $c$  and variance  $O(c/\text{Reps})$ . Thus, for an appropriately large  $\text{Reps}$ , we can apply Chebyshev's inequality to show that  $\bar{Z}^{(k)}$  is indeed very close to  $c$  with high probability. The following proposition makes this intuition formal. Let  $k_0$  be the unique negative power of 2 such that

$$\frac{3}{4} < c_0 = \frac{F_0(\mathcal{S})}{2^{k_0}} \leq \frac{3}{2} \quad (3)$$

► **Proposition 15.** *Let  $\text{Reps} = \lceil 90/\varepsilon^2 \rceil$  and let  $k_0$  be the number defined in Equation 3. For every  $k \geq k_0 - 1$ , let  $Z_1^{(k)}, \dots, Z_{\text{Reps}}^{(k)}$  be the  $\text{Reps}$  many independent  $(F_0(\mathcal{S})/2^k, \varepsilon/2)$ -bucket load random variables. Then,*

1.  $\bar{Z}^{(k_0)}$  and  $\bar{Z}^{(k_0-1)}$  are  $(\varepsilon, 1/24)$ -approximation of  $c_0$  and  $2c_0$  respectively. Hence,  $2^{k_0} \cdot \bar{Z}^{(k_0)}$  and  $2^{k_0-1} \cdot \bar{Z}^{(k_0-1)}$  are  $(\varepsilon, 1/24)$ -approximations of  $F_0(\mathcal{S})$ .
2.  $\Pr(\bar{Z}^{(k_0-1)} < 1) + \sum_{i \geq 1} \Pr(\bar{Z}^{(k_0+i)} \geq 1) \leq 1/4$ .

We postpone the proof of Proposition 15 to the Appendix B.

Item 2 of the Proposition 15 proves that  $k^* \in \{k_0, k_0 - 1\}$  with probability at least  $3/4$  (where  $k^*$  is defined in **line 16**). The first part proves that  $\bar{Z}^{(k_0)} \cdot 2^{k_0}$  and  $\bar{Z}^{(k_0-1)} \cdot 2^{k_0-1}$  are  $(\varepsilon, 1/24)$ -approximations of  $F_0(\mathcal{S})$ . Hence, we observe that  $\bar{Z}^{(k^*)} 2^{k^*}$  is an  $(\varepsilon, 1/3)$ -approximation of  $F_0(\mathcal{S})$  (as this can fail either when  $k^* \notin \{k_0, k_0 - 1\}$ , or  $\bar{Z}^{(k_0)} \cdot 2^{k_0}$  or  $\bar{Z}^{(k_0-1)} \cdot 2^{k_0-1}$  fails to approximate  $F_0(\mathcal{S})$ , the probability of any one of these can happen with probability at most  $1/3$ ). This proves that the Algorithm 2 is an  $F_0$ -estimator for Delphic Sets with WOR sampling.

### 3.1.2 Complexity of the Algorithm 2

The space complexity and the update time complexity are clear from the pseudo-code of the algorithm. The size of  $\mathcal{X}_r^{(k)}$  is upper bounded by  $\text{thresh}$ . And since  $r$  ranges from 1 to  $\text{Reps}$  and  $k$  ranges from 1 to  $\log |\Omega|$ , so the maximum number of elements of  $\Omega$  stored is  $O\left(\frac{\log |\Omega|}{\varepsilon^2}\right)$ . Nevertheless to store an element of  $\Omega$ , one needs  $O(\log |\Omega|)$  bits. So the total space required is  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$ . The additional space required for bookkeeping is  $O(\log |\Omega|)$ .

By the definition of Delphic Sets with WOR sampling, knowing the size of a set  $S_j$  and checking if an element is in  $S_j$  can be done in  $O(\log |\Omega|)$  time. So the for any  $j, r$  and  $k$  the **lines 8 to 9** can be done is at most  $O(|\mathcal{X}_r^{(k)}| \log(|\Omega|)) = O(\text{thresh} \log(|\Omega|))$  step. The number of samples drawn from any set  $S_j$  (in **line 12** is at most  $\text{thresh}$ . So the time taken in **line 12** is also  $O(\text{thresh} \log(|\Omega|))$ . We should note here that we assume **line 10** that is drawing a sample from  $\text{Bin}(|S_j|, 1/2^k)$  can be done in order  $O(\text{thresh} \log(|\Omega|))$  steps, since  $|S_j| \leq |\Omega|$  and  $k \leq \log |\Omega|$ . We have to do these steps for all  $k$  and  $r$ . Thus in total the update time complexity is  $O(\text{thresh} \log(|\Omega|) \cdot (\log |\Omega|) \cdot \text{Reps}) = O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$ .

Thus, we have the Theorem 12.

### 3.2 $F_0$ - Estimator for General Delphic Sets

Algorithm 2 in **line** 12 uses the ability to sample from the sets in the stream with WOR sampling. This property is not available in general Delphic Sets. However, with a little modification to Algorithm 2, we can also have an  $F_0$ -estimator for general Delphic Sets, which is what Theorem 3 states.

**Proof of Theorem 3.** To have an  $F_0$ -estimator for general Delphic sets, we need to make two important changes to Algorithm 2. Firstly, we set the **thresh** to be  $\max\{\lceil \log 4/\varepsilon \rceil, 18\}$  (instead of  $\max\{\lceil \log 2/\varepsilon \rceil, 18\}$ ). Secondly, we change the **line** 12 to the following:

$$12: \mathcal{L}_r^{(k)} = \text{WRSample}(S, d_r^{(k)}, d_r^{(k)} \log d_r^{(k)} \log(\frac{4}{\varepsilon}))$$

While in Algorithm 2 in **line** 12 the set  $|\mathcal{L}_r^{(k)}| = d_r^{(k)}$  with probability 1, in the modified algorithm this is not the case. In other words, if we try to prove the correctness of the modified algorithm in the same line as the proof of Algorithm 2 we note that the random variable  $Y_{i,r}^{(k)}$  takes value slightly differently.

The random variable  $Y_{i,r}^{(k)}$  can take value 1 for two possible cases.

1. Due to the overflow of the bucket, which can now happen with probability at most  $2^{-\text{thresh}} \leq \varepsilon/4$  (This is due to the modified setting of **thresh** so that  $2^{-\text{thresh}} \leq \varepsilon/4$ ).
2. Due to **WRSample** returning an empty set. By Lemma 11 and by the setting of the inputs of **WRSample** this can happen also with probability at most  $\varepsilon/4$ .

Thus we can prove a lemma corresponding to Lemma 13 and hence,  $Z_r^{(k)}$  is  $(c, \alpha)$ -bucket-load where  $\alpha = \min\{2^{-18}, \varepsilon/2\}$ . Now, we can apply our core Proposition 15 to conclude the correctness of the algorithm.

It is easy to see that the space and time complexity of the modified algorithm is  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$  and  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log^3 \varepsilon^{-1}\right)$  respectively.  $\blacktriangleleft$

## 4 Space Optimal Algorithm with an Oracle in NTISP(poly, LINSPEACE)

In this section, we give a sketch of the proof of Theorem 4. We will implement the  $F_0$ -estimation algorithm by Gibbons and Tirthapura [13] for singleton streams as described in [2]. The proof that the algorithm gives the correct estimation follows from their proof and is hence omitted. The algorithm picks a random hash function  $h$  from a pairwise-independent family and keeps a bucket  $\mathcal{X}$  of hash values of a subset of elements in the stream and a number  $z$  of leading zeros of all the elements in the bucket. The size of the bucket is bounded by  $O(\frac{1}{\varepsilon^2})$ . When a new item  $x$  comes, if the number of leading zeros of  $h(x) \geq z$ , the algorithm updates the bucket to  $\mathcal{X} \cup \{h(x)\}$ . If the bucket overflows, then  $z$  is updated to  $z + 1$ , and all elements from  $\mathcal{X}$  with leading zeros  $\leq z$  are removed. At the end of the stream, the algorithm outputs  $|\mathcal{X}| \cdot 2^z$  as the estimate. In adapting their algorithm for the case of set streams, the computational challenge is implementing the update. The central intuition is that this can be accomplished by querying a language in NTISP(poly, LINSPEACE) with query size  $O(\log |\Omega| \cdot \frac{1}{\varepsilon^2})$ . Thus if NTISP(poly, LINSPEACE) = DTISP(poly, LINSPEACE), then queries to the language can be simulated in space  $O(\log |\Omega| \cdot \frac{1}{\varepsilon^2})$  and time  $\text{poly}(\log |\Omega|, \varepsilon^{-1})$ .

We will use  $n$  to denote  $\log |\Omega|$ , the length of the representation of any element in the universe  $\Omega$ . For any binary string  $y$ ,  $\text{Zero}(y)$  denotes the number of leading zeros of  $y$ . We will use the standard Toeplitz family of pairwise-independent hash functions denoted by  $H_{\text{Teop}}(n, k)$  (see Appendix D for details). Let  $h$  be a hash function. For every  $m \in \{1, \dots, n\}$ ,

## 26:14 Improved Streaming Algorithm for the Klee's Measure Problem and Generalizations

the  $m^{\text{th}}$  prefix-slice of  $h$ , denoted  $h_m$ , is a map from  $\{0, 1\}^n$  to  $\{0, 1\}^m$ , where  $h_m(y)$  is the first  $m$  bits of  $h(y)$ . If  $h$  is from  $\mathbf{H}_{\text{Teop}}(n, k)$ , both  $h$  and  $h_m$  are efficiently computable and take linear space to represent.

For a member  $S \subseteq \Omega$  of the Delphic set, we will assume a representation of size  $O(|\Omega|)$  and denote it by  $r_S$ . This is required for computational purposes, and all the Delphic families discussed in earlier works have such representations (e.g., rectangles in KMP). We use the notation  $x \in r_S$  to mean  $x \in S$ . Note that since  $S$  is a Delphic set, membership can be checked in time and space linear in the representation.

### Oracle Languages

We will define two languages,  $L$  and  $L_{\text{pref}}$ , and show that they are in  $\text{NTISP}(\text{poly}, \text{LINSPEACE})$ . We use these languages to implement Gibbons and Tirthapura algorithms. The first language,  $L$ , checks whether the bucket overflows by adding new elements from the current set  $S$  with the current value of the leading zeros count.

$$L = \{ \langle r_S, h, \mathcal{X}, \ell, m, 1^{\ell \log n} \rangle \mid \exists x_1 < x_2 < \dots < x_k \text{ such that } x_i \in r_S \\ \& \forall i (h_m(x_i) = 0^m) \& |\{h(x_1), \dots, h(x_k)\} \cup \mathcal{X}| > \ell \}.$$

The following language is (close to) the *prefix language* of  $L$  that can be used to construct witnesses  $x_1, \dots, x_k \in r_S$  if adding new elements does not result in an overflow of  $\mathcal{X}$ .

$$L_{\text{pref}} = \{ \langle r_S, h, \mathcal{X}, \ell, m, 1^{\ell \log n}, i, j \rangle \mid \exists x_1 < x_2 < \dots < x_k \text{ such that} \\ x_i \in r_S \& \forall i (h_m(x_i) = 0^m) \& |\{h(x_1), \dots, h(x_k)\} \cup \mathcal{X}| \leq \ell \& i^{\text{th}} \text{ bit of } x_j \text{ is } 1 \}.$$

To implement the algorithm, we must update  $\mathcal{X}$  with new elements from  $S$  (represented by  $r_S$ ), the current set in the stream. For this, the algorithm uses a subroutine **Construct** that in turn uses  $L_{\text{pref}}$  to search for a set of  $\leq l$  the elements  $x_i$ s in  $S$  so that  $h_m(x_i) = 0^m$ . We will ensure that we will use **Construct** only when we are guaranteed that adding the hash values on these elements will not make  $\mathcal{X}$  overflow. For this, we will use membership in  $L$ . The Algorithm 3 is described below. Theorem 4 follows from Claim 16, Claim 17, and the guarantee of Gibbons and Tirthapura's algorithm.

#### ■ Algorithm 3 DelphicWithNP( $S, \varepsilon$ ).

---

```

1:  $h \xleftarrow{R} \mathbf{H}_{\text{Teop}}(n, n)$ 
2:  $\mathcal{X} \leftarrow \phi$ ,  $\text{Const} \leftarrow \frac{100}{\varepsilon^2}$ ,  $m \leftarrow 0$ 
3: while not End-of-Stream do
4:   On item  $r_S$ 
5:   while  $\langle r_S, h, \mathcal{X}, \text{Const}, m, 1^{\text{Const} \cdot \log n} \rangle \notin L$  do
6:     Remove all  $y$ s from  $\mathcal{X}$  for which  $\text{Zero}(y) = m$ 
7:      $m \leftarrow m + 1$ 
8:    $\mathcal{X} \leftarrow \mathcal{X} \cup \text{Construct}(\langle r_S, h, \mathcal{X}, \text{Const}, 1^{\text{Const} \cdot \log n} \rangle)$ 
9: Output  $|\mathcal{X}|2^m$ 

```

---

▷ Claim 16. Both  $L$  and  $L_{\text{pref}}$  are in  $\text{NTISP}(\text{poly}, \text{LINSPEACE})$ .

Proof. We show membership of  $L$  in  $\text{NTISP}(\text{poly}, \text{LINSPEACE})$ . The proof of membership of  $L_{\text{pref}}$  is similar. Consider the following non-deterministic machine  $M_L$ .  $M_L$  on input  $\langle r_S, h, \mathcal{X}, 1^\ell, m \rangle$  first guesses a number  $1 \leq k \leq l$  and  $x_1 < x_2 < \dots < x_k$  in  $\Omega$ . Then it

verifies the following: (1)  $\forall i x_i \in r_S$ , (2)  $\forall i h_m(x_i) = 0^m$ , (3)  $|\{h(x_1), \dots, h(x_k)\} \cup \mathcal{X}| > \ell$ . The input size is  $O(|\mathcal{X}| + \ell \cdot \log n)$ . Since all the steps: guessing and verification (1), (2), and (3), can be done in time polynomial in the input length,  $M_L$  runs in polynomial time. The critical observation is that  $M_L$  can write down all the guesses in space linear in the input (at most  $\ell$   $x_i$ s from  $\Omega$  that takes  $O(\ell \log n)$  bits to store). Thus, the overall space used is linear in the input length.  $\triangleleft$

Let  $k$  be the maximum value of  $|\{h(x_1), \dots, h(x_t)\} \cup \mathcal{X}|$  so that the following property  $\mathcal{P}$  holds:

1.  $\exists x_1 < x_2 < \dots < x_t$  such that  $x_i \in r_S$
2.  $\forall i (h_m(x_i) = 0^m)$ .

$\triangleright$  **Claim 17.** There is a deterministic algorithm **Construct**, takes  $\langle r_S, h, \mathcal{X}, \text{Const}, 1^{\text{Const} \cdot \log n} \rangle$  as input and  $L$  and  $L_{\text{pref}}$  as oracles and if  $k \leq \text{Const}$ , it returns a set  $\{h(x_1), \dots, h(x_k)\}$  (if non-empty) so that  $\forall i (h_m(x_i) = 0^m)$  and  $x_i \in r_S$ . **Construct** runs in time polynomial and space linear in the input length and makes only queries that are linear in the input length.

**Proof (Sketch).** **Construct** first queries  $L$  to check  $k > \text{Const}$ . If  $k > \text{Const}$ , it rejects. Otherwise, it first finds  $k$  and uses  $k \log n$  queries to  $L_{\text{pref}}$  to construct all  $x_i$ s in  $S$  with the property  $\mathcal{P}$ . The complexity bounds are clear from the language's description and definition.  $\triangleleft$

## 5 Conclusion

In this paper, we present a new elegant algorithm that improves both the space and update time complexities for the computation of the size of the union of Delphic sets. The space and time complexities of our algorithm are  $\tilde{O}\left(\log^2(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2}\right)$ . To complement our algorithm, we also show that, given access to  $\text{NTISP}(\text{poly}, \text{Linspace})$  oracle, there exists a *hashing* based algorithm for approximating the size of the union of Delphic sets with space complexity  $O(\log(|\Omega|) \cdot \varepsilon^{-2} \cdot \delta^{-1})$  and  $\text{poly}(\log(|\Omega|), \varepsilon^{-1}, \log \frac{1}{\delta})$  update time complexity. Note  $\Omega(\log(|\Omega|))$  lower bound for our problem follows directly from the lower bound of  $F_0$ -estimation problem, but the above result implies that if the complexity of our problem is  $\omega(\log(|\Omega|))$ , then we should not expect this to be proved soon.

---

## 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. doi:10.1006/jcss.1997.1545.
- 2 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proc. of RANDOM*, pages 1–10, 2002. doi:10.1007/3-540-45726-7\_1.
- 3 Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Phys. Rev. E*, 71:036113, March 2005. doi:10.1103/PhysRevE.71.036113.
- 4 Jon Louis Bentley. Algorithms for klee's rectangle problems. Technical report, Technical Report, Computer, 1977.
- 5 Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. In *Proc. of SODA*, 2018. doi:10.1137/1.9781611975031.156.
- 6 Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.*, 43(6-7):601–610, 2010. doi:10.1016/j.comgeo.2010.03.004.

- 7 Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. doi:10.1016/0022-0000(79)90044-8.
- 8 Timothy M. Chan. A (slightly) faster algorithm for klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. doi:10.1016/j.comgeo.2009.01.007.
- 9 Eric Y Chen and Timothy M Chan. Space-efficient algorithms for klee's measure problem. *algorithms*, 3(5):6, 2005.
- 10 Bogdan S. Chlebus. On the klee's measure problem in small dimensions. In Branislav Rován, editor, *SOFSEM '98: Theory and Practice of Informatics, 25th Conference on Current Trends in Theory and Practice of Informatics, Jasná*, volume 1521, pages 304–311, 1998. doi:10.1007/3-540-49477-4\_22.
- 11 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985. doi:10.1016/0022-0000(85)90041-8.
- 12 Michael L Fredman and Bruce Weide. On the complexity of computing the measure of  $\bigcup [a_i, b_i]$ . *Communications of the ACM*, 21(7):540–544, 1978.
- 13 E. Grädel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Lecture Notes in Computer Science 2500. Springer, 2002.
- 14 Joachim Gudmundsson and Rasmus Pagh. Range-efficient consistent sampling and locality-sensitive hashing for polygons. In *28th International Symposium on Algorithms and Computation, ISAAC*, volume 92 of *LIPICs*, pages 42:1–42:13, 2017. doi:10.4230/LIPICs.ISAAC.2017.42.
- 15 Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 283–288. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238202.
- 16 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS*, pages 41–52, 2010. doi:10.1145/1807085.1807094.
- 17 Victor Klee. Can the measure of be computed in less than  $o(n \log n)$  steps? *The American Mathematical Monthly*, 84(4):284–285, 1977.
- 18 Kuldeep S. Meel, Sourav Chakraborty, and N. V. Vinodchandran. Estimation of the size of union of delphic sets: Achieving independence from stream size. In Leonid Libkin and Pablo Barceló, editors, *PODS*, pages 41–52. ACM, 2022. doi:10.1145/3517804.3526222.
- 19 Kuldeep S. Meel, N. V. Vinodchandran, and Sourav Chakraborty. Estimating the size of union of sets in streaming models. In *Proc. of PODS*, pages 126–137, 2021. doi:10.1145/3452021.3458333.
- 20 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.
- 21 Mark H Overmars and Chee-Keng Yap. New upper bounds in klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991. doi:10.1137/0220065.
- 22 A. Pavan and Srikanta Tirthapura. Range-efficient counting of distinct elements in a massive data stream. *SIAM J. Comput.*, 37(2):359–379, 2007. doi:10.1137/050643672.
- 23 Aduri Pavan, N. V. Vinodchandran, Arnab Bhattacharya, and Kuldeep S. Meel. Model counting meets  $f_0$  estimation. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 299–311. ACM, 2021. doi:10.1145/3452021.3458311.
- 24 He Sun and Chung Keung Poon. Two improved range-efficient algorithms for  $f_0$  estimation. *Theor. Comput. Sci.*, 410(11):1073–1080, 2009. doi:10.1016/j.tcs.2008.10.031.
- 25 Srikanta Tirthapura and David P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In *Proc. of PODS*, pages 283–294. ACM, 2012. doi:10.1145/2213556.2213595.
- 26 Richard Ryan Williams. Time-space tradeoffs for counting np solutions modulo integers. *computational complexity*, 17:179–219, 2007. URL: <https://api.semanticscholar.org/CorpusID:8815358>.

## A Concentration Bounds

► **Lemma 18.** *Let  $Z$  be a random variable with  $c(1 - \varepsilon/2) \leq \text{Ex}(Z) \leq c$  for some  $c > 0$  and  $0 \leq \varepsilon < 1$ . Then,*

$$\Pr(Z \neq (1 \pm \varepsilon)c) \leq \frac{4 \text{Var}(Z)}{c^2 \varepsilon^2}.$$

Note that the lower tail event  $Z \leq c - c\varepsilon$  implies  $Z - \text{Ex}(Z) \leq c\varepsilon/2$ . Thus,  $|Z - c| > c\varepsilon$  implies that  $|Z - \text{Ex}(Z)| > c\varepsilon/2$ . Thus, the above result follows from the Chebyshev's inequality. The following lemma directly follows from Chebyshev's inequality.

► **Lemma 19.**

1. *Let  $Z$  be a random variable with  $\text{Ex}(Z) \geq \mu$ . Then,*

$$\Pr(Z \leq \mu - t) \leq \frac{\text{Var}(Z)}{t^2}.$$

2. *Let  $Z$  be a random variable with  $\text{Ex}(Z) \leq \mu$ . Then,*

$$\Pr(Z \geq \mu + t) \leq \frac{\text{Var}(Z)}{t^2}.$$

We will also need the following Chernoff bound.

► **Lemma 20** (See Theorem 4.4 from [20]). *Suppose  $T \sim \text{Bin}(n, p)$  and  $L \geq 6np$  then*

$$\Pr(T \geq L) \leq 2^{-L}.$$

## B Proof of Proposition 15

We first observe the simple properties of the first and second moments of averages of the independent bucket-loads.

► **Lemma 21.** *Let  $c_0$  and  $k_0$  be defined as in Equation 3. Then,*

- $c_0(1 - \varepsilon/2) \leq \mathbb{E}(\bar{Z}^{(k_0)}) \leq c_0$ ,
- $\text{Var}(\bar{Z}^{(k_0)}) \leq V_0 := \frac{c_0 + c_0^2 2^{-18}}{\text{Reps}} \leq \frac{1}{80}$ .
- $\text{Var}(\bar{Z}^{(k_0-1)}) \leq V_{-1} := \frac{2c_0 + 4c_0^2 \varepsilon/8}{\text{Reps}} \leq 2V_0 \leq \frac{1}{40}$ .
- For all  $i \geq 1$ ,

$$\text{Var}(\bar{Z}^{(k_0+i)}) \leq V_i := \frac{2^{-i} \cdot c_0 + 2^{-2i} c_0^2 \varepsilon/2}{\text{Reps}} \leq V_0/2^i$$

The above follows directly from Lemma 13 along with the choice of  $\text{Reps} = 99/\varepsilon^2$  and  $\varepsilon \leq 0.5$ . Now, by using Lemma 18,

$$\Pr(\bar{Z}^{(k_0)} \neq (1 \pm \varepsilon)c_0) \leq \frac{4\text{Var}(\bar{Z}^{(k_0)})}{c_0^2 \varepsilon^2} \leq \frac{2}{\varepsilon^2 \text{Reps}} \left( \frac{2}{c_0} + \varepsilon \right) \leq \frac{(8/3) + 2\varepsilon}{\varepsilon^2 \text{Reps}} \leq 1/24.$$

The last inequality follows from the choice of  $\text{Reps} = 90/\varepsilon^2$  and we assume that  $\varepsilon \leq 0.5$ . So,  $\bar{Z}^{(k_0)}$  is an  $(\varepsilon, 1/24)$ -approximation of  $c_0$ . Similarly, by using Lemma 18,  $\bar{Z}^{(k_0-1)}$  is an  $(\varepsilon, 1/24)$ -approximation of  $2c_0$ . This completes the proof of Item 1 in Proposition 15.

Now we prove the Item 2 of Proposition 15.

## 26:18 Improved Streaming Algorithm for the Klee's Measure Problem and Generalizations

We start by bounding  $\Pr(\bar{Z}^{(k_0-1)} < 1)$ . Note that  $\mathbb{E}(\bar{Z}^{(k_0-1)}) \geq 2c_0(1 - 2^{-\text{thresh}}) \geq 3/2 \times (1 - 2^{-18}) \geq 3/2 - \beta$ , where  $\beta$  is a negligible real number such that  $3/2^{19} \leq \beta \leq 2^{-17}$ . Now it is easy to see that there exists some  $\alpha \in \mathbb{R}$  with  $2^{-16} \leq \alpha \leq 2^{-15}$  and  $\alpha^2 + 4\alpha < 1$  such that  $\frac{1}{2+\alpha} \leq 1/2 - \beta$ . By using Lemma 19,

$$\Pr(\bar{Z}^{(k_0-1)} < 1) \leq (2 + \alpha)^2 \text{Var}(\bar{Z}^{(k_0-1)}) = (4 + 4\alpha + \alpha^2) V_{-1} \leq 5V_{-1} = 10V_0 \quad (4)$$

Now let us bound  $\Pr(\bar{Z}^{(k_0+i)} \geq 1)$  for every  $i \geq 1$ . Note that  $\mathbb{E}(\bar{Z}^{(k_0+i)}) \leq c_0/2 \leq 3/4$ . So, by Lemma 19 we obtain

$$\Pr(\bar{Z}^{(k_0+i)} \geq 1) \leq 16V_1 = 8V_0 \quad (5)$$

Note that for  $i \geq 2$  we have  $\mathbb{E}(\bar{Z}^{(k_0+i)}) \leq c_0/2^i \leq 1/2$ . Applying Lemma 19 working out in detail we obtain,

$$\Pr(\bar{Z}^{(k_0+i)} > 1) \leq 4\text{Var}(\bar{Z}^{(k_0+i)}) \leq 2^{-i+2}V_0$$

Thus using the infinite geometric sum we obtain

$$\sum_{i \geq 2} \Pr(\bar{Z}^{(k_0+i)} \geq 1) \leq 2V_0 \quad (6)$$

Hence from the inequalities 5 and 6 we obtain

$$\sum_{i \geq 1} \Pr(\bar{Z}^{(k_0+i)} \geq 1) \leq 10V_0 \quad (7)$$

Finally, combining the inequalities 4 and 7 we conclude

$$\Pr(\bar{Z}^{(k_0-1)} < 1) + \sum_{i \geq 1} \Pr(\bar{Z}^{(k_0+i)} \geq 1) \leq 1/4$$

This completes the proof of Item 2 of Proposition 15.  $\blacktriangleleft$

## C Further Improvements for Klee's Measure Problem

Recall, every  $d$ -dimensional rectangle is a Delphic set with  $\Omega = [\Delta]^d$ . Therefore, Theorem 3 provides an algorithm that has space complexity  $O\left(\frac{d^2 \log^2 \Delta}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$  and has update time complexity  $O\left(\frac{d^2 \log^2 \Delta}{\varepsilon^2} \cdot \log^3 \varepsilon^{-1}\right)$  for Klee's Measure Problem. We now discuss how we can further improve the dependence on  $\varepsilon$ . The key idea behind such an improvement is to observe that we can avoid the overhead due to WOR by replacing with sampling from Geometric distribution. Such a replacement is possible only because there exists a total ordering for all the elements in  $\Omega = [\Delta]^d$  such that we can access  $i$ -th element in time  $\log(|\Omega|)$ . Interestingly, all the known classes of Delphic sets satisfy the above property and therefore, we formalize such a family below:

► **Definition 22.** *A set  $S \subseteq \Omega$  belongs to Ordered Delphic family if there exists a total ordering  $\leq$  for all the elements in the  $\Omega$  and the following queries can be done in  $O(\log |\Omega|)$  time.*

1. Know the size of the set  $S$ ,
2. For any  $1 \leq i \leq |S|$  the  $i$ th element of the set  $S$  can be obtained,
3. Given any  $x$  check if  $x \in S$ .

In Section 2.4 we discussed how one can sample a Bernoulli process from a Delphic Set by using the coupon collector theorem. There is an alternate way to sample a Bernoulli process if the set is a Ordered Delphic Set. The set  $L$  returned by the algorithm  $Sample(S, p)$  follows  $Ber(S, p)$ . Since there is an order to the elements in a Ordered Delphic set, the process is basically same as sampling a Bernoulli process from  $[|S|]$ . This is a well known result in probability theory and details can be found in [3].

■ **Algorithm 4**  $GeometricSample(n, p)$ .

---

```

1: Input Set  $n$   $p \in [0, 1]$ .
2: Initialize  $L = \emptyset$ ;  $t = 0$ 
3: for  $t \leq |S|$  do
4:   Sample  $g$  from  $Geo(p)$ 
5:    $t = t + g$ 
6:   if  $t \leq n$  then  $L = L \cup \{t\}$ 
7: Output  $L$ 

```

---

► **Lemma 23.** *The subroutine  $GeometricSample(n, p)$  outputs a subset of  $[n]$  according to the Bernoulli process. More precisely, if we set  $I_j = 1$  if  $j \in L$ , 0 otherwise then  $I_1, \dots, I_n$  follow identically and independently distributed to  $Ber(p)$ .*

Using the subroutine  $GeometricSample$  we can now present the improved algorithm (Algorithm 5) for the Ordered Delphic Sets.

■ **Algorithm 5** [F<sub>0</sub>-Estimator for Ordered Delphic Sets.]

---

```

1: Input Stream =  $\langle S_1, S_2, \dots, S_M \rangle$ ,  $\varepsilon, \delta$ 
2: Initialize
3:  $p \leftarrow 1$ ; thresh  $\leftarrow \max\{\lceil \log 2/\varepsilon \rceil, 18\}$ ; Reprs  $\leftarrow \lceil 90/\varepsilon^2 \rceil$ ;
4: for  $(1 \leq k \leq \log n)$  and  $(1 \leq r \leq \text{Reprs})$  do
5:    $\mathcal{X}_r^{(k)} \leftarrow \emptyset$ ;

   STREAMING PHASE:
6: for  $j = 1$  to  $M$  do
7:   for  $(1 \leq k \leq \log n)$  and  $(1 \leq r \leq \text{Reprs})$  do
8:     for all  $s \in \mathcal{X}_r^{(k)}$  do
9:       if  $s \in S_j$  then remove  $s$  from  $\mathcal{X}_r^{(k)}$ 
10:    Draw  $D_r^{(k)}$  from  $GeometricSample(|S_j|, 1/2^k)$ 
11:    if  $|\mathcal{X}_r^{(k)}| + |D_r^{(k)}| \leq \text{thresh}$  then
12:       $\mathcal{L}_r^{(k)} = \{S_j[\ell] \mid \ell \in D_r^{(k)}\}$ 
13:       $\mathcal{X}_r^{(k)} = \mathcal{X}_r^{(k)} \cup \mathcal{L}_r^{(k)}$ 

   POST-STREAMING PHASE:
14: for  $1 \leq k \leq \log n$  do
15:    $\bar{z}^{(k)} = \frac{1}{\text{Reprs}} \sum_{r=1}^{\text{Reprs}} |\mathcal{X}_r^{(k)}|$ 
16:  $k^* = \max\{k \in [1, \log n] : \bar{z}^{(k)} \geq 1\}$ 
17: Output  $\bar{z}^{(k^*)} 2^{k^*}$ 

```

---

Algorithm 5 is the streaming algorithm for the Ordered Delphic Sets. For proving the correctness of Algorithm 5, we have the following theorem:



► **Theorem 24** (correctness theorem of Algorithm 5). *If  $\mathcal{S}$  is a stream of Ordered Delphic Set the Algorithm 5 is an  $(\varepsilon, 1/3)$ - $F_0$ -estimator. Also, the space and update time complexity of Algorithm 5 is  $O\left(\frac{\log^2 |\Omega|}{\varepsilon^2} \cdot \log \varepsilon^{-1}\right)$ .*

**Proof of Theorem 24.** Note that, the only difference between Algorithm 5 and Algorithm 2 is in **lines** 10, 11 and 12. Also note that the Algorithm 5 behaves identically if **lines** 10 to 13 is replaced by

- 10: Draw  $D_r^{(k)}$  from  $\text{GeometricSample}(|S_j|, 1/2^k)$
- 11:  $\mathcal{L}_r^{(k)} = \{S_j[\ell] \mid \ell \in D_r^{(k)}\}$
- 12: **if**  $|\mathcal{X}_r^{(k)}| + |D_r^{(k)}| \leq \text{thresh}$  **then**
- 13:      $\mathcal{X}_r^{(k)} = \mathcal{X}_r^{(k)} \cup \mathcal{L}_r^{(k)}$

By Lemma 23 the size of  $D_r^{(k)}$  is distributed according to the binomial distribution  $\text{Bin}(|S_k|, 1/2^k)$ , that is the distribution of  $d_r^{(k)}$  in Algorithm 2 is same as the distribution of  $|D_r^{(k)}|$  in Algorithm 5. Thus,  $L_r^{(k)}$  is distributed according to  $\text{Ber}(S_j, 1/2^k)$  which is also the same distribution if  $|D_r^{(k)}|$  samples are drawn from  $S_j$  using WOR sampling.

Thus the correctness of the Algorithm 5 follows from the correctness of Algorithm 2. The complexities of Algorithm 5 is also identical to that of Algorithm 2. ◀

Recall that every  $d$ -dimensional rectangle can be viewed as an Ordered Delphic set, therefore, Theorem 24 implies the following result in the context of Klee's Measure Problem.

► **Corollary 25** (Improved Algorithm for KMP). *There is a streaming algorithm for the Klee's Measure Problem with space and update time complexity  $O\left(\frac{d^2 \log^2 \Delta}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right)$ .*

## D Basic Probability Results



► **Definition 26** (Geometric Distribution). *Given any  $p \in (0, 1]$  the Geometric Distribution over the set of positive integers  $\{1, 2, 3, \dots\}$  is denoted as  $\text{Geo}(p)$  where probability of a positive integer  $k$  is  $(1 - p)^{k-1}p$ .*

► **Theorem 27** (Coupon Collection Problem). *Given access to uniform random samples from a set  $T$  and a number  $r \leq |T|$ , let  $W_r$  be a random variable that stand for the number of independent uniform random samples from  $T$  needed before we get  $r$  distinct samples from  $T$ . Then  $\Pr(W_r > \beta r \log r) \leq r^{-\beta+1}$ .*

► **Remark 28** (Pairwise Independent Hash Function). We will use pairwise independent hash functions. For an integer  $n, k$  and  $\mathcal{H}(n, k) \triangleq \{h : \{0, 1\}^n \rightarrow \{0, 1\}^k\}$  be a family of hash functions mapping  $\{0, 1\}^n$  to  $\{0, 1\}^k$ . We use  $h \xleftarrow{R} \mathcal{H}(n, k)$  to denote the probability space obtained by choosing a function  $h$  uniformly at random from  $\mathcal{H}(n, k)$ . A family of hash functions  $\mathcal{H}(n, k)$  is 2-wise independent if  $\forall \alpha_1, \alpha_2 \in \{0, 1\}^k$ , distinct  $x_1, x_2 \in \{0, 1\}^n$ ,  $h \xleftarrow{R} \mathcal{H}(n, k)$ ,  $\Pr[(h(x_1) = \alpha_1) \wedge (h(x_2) = \alpha_2)] = \frac{1}{2^{2k}}$ . Several families of 2-wise independent hash functions are known. We will use  $\text{H}_{\text{Teop}}(n, k)$ , which is known to be 2-wise independent [7]. The family is defined as follows:  $\text{H}_{\text{Teop}}(n, k) \triangleq \{h : \{0, 1\}^n \rightarrow \{0, 1\}^k\}$  is the family of functions of the form  $h(x) = Ax + b$  with  $A \in \mathbb{F}_2^{k \times n}$  and  $b \in \mathbb{F}_2^{k \times 1}$  where  $A$  is a uniformly randomly chosen Toeplitz matrix of size  $k \times n$  while  $b$  is uniformly randomly matrix of size  $k \times 1$ . It is worth noticing that  $\text{H}_{\text{Teop}}$  can be represented with  $\Theta(n)$ -bits and hash value of any element  $x \in \{0, 1\}^n$  can be computed in  $O(nk)$  time.



# An EPTAS for Cardinality Constrained Multiple Knapsack via Iterative Randomized Rounding

Ilan Doron-Arad  

Computer Science Department, Technion, Haifa, Israel

Ariel Kulik  

Computer Science Department, Technion, Haifa, Israel

Hadas Shachnai  

Computer Science Department, Technion, Haifa, Israel

---

## Abstract

In [Math. Oper. Res., 2011], Fleischer et al. introduced a powerful technique for solving the generic class of *separable assignment problems* (SAP), in which a set of items of given values and weights needs to be packed into a set of bins subject to separable assignment constraints, so as to maximize the total value. The approach of Fleischer et al. relies on solving a configuration LP and sampling a configuration for each bin independently based on the LP solution. While there is a SAP variant for which this approach yields the best possible approximation ratio, for various special cases, there are discrepancies between the approximation ratios obtained using the above approach and the state-of-the-art approximations. This raises the following natural question: Can we do better by *iteratively* solving the configuration LP and sampling a few bins at a time?

To assess the potential of the iterative approach we consider a specific SAP variant as a case-study, UNIFORM CARDINALITY CONSTRAINED MULTIPLE KNAPSACK, for which we answer this question affirmatively. The input is a set of items, each has a value and a weight, and a set of uniform capacity bins. The goal is to assign a subset of the items of maximum total value to the bins such that (i) the capacity of any bin is not exceeded, and (ii) the number of items assigned to each bin satisfies a given *cardinality* constraint. While the technique of Fleischer et al. yields a  $(1 - \frac{1}{e})$ -approximation for the problem, we show that iterative randomized rounding leads to *efficient polynomial time approximation scheme* (EPTAS), thus essentially resolving the complexity status of the problem. Our analysis of iterative randomized rounding may be useful for solving other SAP variants.

**2012 ACM Subject Classification** Theory of computation

**Keywords and phrases** multiple knapsack, cardinality constraint, EPTAS, iterative randomized rounding

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.27

**Category** APPROX

**Related Version** *Full Version:* <https://doi.org/10.48550/arXiv.2308.12622> [9]

**Funding** *Ariel Kulik:* This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 852780-ERC (SUBMODULAR)

## 1 Introduction

We consider problems in the class of maximizing assignment problems with packing constraints, also known as SEPARABLE ASSIGNMENT PROBLEMS (SAP). A general problem in this class is defined by a set of bins and a set of items to be packed in the bins. There is a *value*  $v_{ij}$  (also called *profit* sometimes) associated with assigning item  $i$  to bin  $j$ . We are also given a separate packing constraint for each bin  $j$ . The goal is to find an assignment of a subset of the items to the bins which maximizes the total value accrued. This class includes several well studied problems such as the GENERALIZED ASSIGNMENT PROBLEM (GAP).



© Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 27; pp. 27:1–27:17



Leibniz International Proceedings in Informatics

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

In [12], Fleischer et al. introduced a powerful technique for solving SAP and its variants. The technique relies on first solving a configuration linear programming (configuration-LP) relaxation of the problem. Subsequently, configurations (i.e., feasible subsets of items for a single bin) are sampled independently according to a distribution specified by the LP solution to obtain an integral solution for the given instance. For many SAP variants, such as GAP, the approximation guarantee of the resulting algorithm is  $(1 - 1/e)$ .

Intuitively, we can do better using the following iterative randomized rounding approach: Iteratively solve a configuration LP relaxation of the problem for the remaining items and bins and sample a *few* configurations based on the distribution specified by the LP solution, until all bins are used. We note that if the LP is solved only once and all of the bins are packed based on the solution, then we have exactly the algorithm of Fleischer et al. [12]. This raises the following question:

Can iterative randomized rounding improve the approximation ratio of [12]?

As shown in [12], under standard complexity assumptions, there is a SAP variant for which their approximation ratio of  $(1 - 1/e)$  is the best possible. However, for various special cases (such as MULTIPLE KNAPSACK and GAP), there are discrepancies between the approximation guarantee obtained using the algorithm of [12] and the state-of-the-art approximations. This indicates that the iterative approach may potentially lead to improved approximation for some variants (compared to [12]). Hence, to assess the potential of the iterative approach we focus as a case study on one interesting SAP variant, namely, UNIFORM CARDINALITY CONSTRAINED MULTIPLE KNAPSACK (CMK). Specifically, we show that iterative randomized rounding is superior to the technique of [12] and use it to essentially resolve the complexity status of this problem.

An input for CMK consists of a set of items, each has a value and a weight, and a set of uniform capacity bins. The goal is to assign a subset of the items of maximum total value to the bins such that (i) the total weight of items in each bin does not exceed its capacity, and (ii) the number of items assigned to each bin satisfies a given cardinality constraint.<sup>1</sup> CMK has real-world applications in cloud computing, as well as in manufacturing systems and radio networks (see the full version of the paper [9]).

## 1.1 Related Work

Iterative randomized rounding of configuration-LPs has been recently used for obtaining the current state-of-the-art approximation for *vector bin packing* in [17]. In this problem, the goal is to pack a set of items, each given by a  $d$ -dimensional size vector for some  $d > 1$ , in a minimum number of  $d$ -dimensional bins, where a subset of items fits in a bin if it adheres to the capacity constraints in all dimensions. We are not aware of an application of iterative randomized rounding of configuration-LPs for *maximization* problems.

The MULTIPLE KNAPSACK WITH UNIFORM CAPACITIES (UMK) problem is the special case of CMK with no cardinality constraint, or equivalently, where the cardinality constraint is larger than the total number of items. In the more general MULTIPLE KNAPSACK (MK) problem, the capacity of the bins may be arbitrary. In terms of approximation algorithms, UMK and MK are well understood. A *polynomial time approximation scheme* PTAS for

---

<sup>1</sup> See a more formal definition in Section 1.3.

UMK was given by Kellerer [16]. Later, Chekuri and Khanna [5] developed the first PTAS for MK and ruled out the existence of a *fully* PTAS (FPTAS), already for UMK with only two bins. Jansen designed more involved *efficient* PTAS (EPTAS) for MK [13, 14], thus resolving the complexity status of the problem.<sup>2</sup>

For CMK, a randomized  $(1 - 1/e)$ -approximation follows from the previously mentioned results of Fleischer et al. [12] for SAP. More specifically, the authors present a randomized algorithm for SAP whose approximation guarantee is  $((1 - 1/e) \cdot \beta)$ , where  $\beta$  is the best approximation ratio for the single bin subproblem.<sup>3</sup> A slightly more efficient approximation ratio for CMK follows from a recent result of Cohen et al. [7] who give a randomized  $(1 - \ln(2)/2 - \varepsilon) \approx 0.653$ -approximation for *uniform 2-dimensional vector multiple knapsack*. In this problem, the cardinality constraint of CMK is generalized to a second knapsack constraint.

We note that a PTAS for CMK can be obtained using ideas of [5]. We outline the main steps. First, item values are discretized into  $O(\log n)$  value classes, where  $n$  is the number of items. Then, enumeration is used to roughly determine the number of items taken from each value class. Clearly, one should take from each value class the items with smallest weights, leading to a reduced problem of packing sufficient items from each value class in the  $m$  given (uniform) bins. Packing these items in  $(1 + \varepsilon) \cdot m + O(1)$  uniform bins can be done using an asymptotic FPTAS for bin packing with cardinality constraint [11] (or more generally, for bin packing with a partition matroid [8]). Finally, the algorithm keeps the  $m$  bins with highest values. We note that the running time of the enumeration step is very high. This leaves open the question whether CMK admits an EPTAS.

## 1.2 Our Results

Our main contribution is in showing that iterative randomized rounding can substantially improve the approximation guarantee of the configuration-LP rounding approach of [12]. The analysis is based on concentration bounds; thus, our iterative algorithm is applied to a slightly restricted subclass of CMK instances in which the value of each configuration is relatively small, and the number of bins is large w.r.t. the given error parameter. Recall that even for two bins the problem does not admit an FPTAS [5]. Hence, we do not expect the iterative approach to work for a small number of bins. More specifically, given an error parameter  $\varepsilon \in (0, 0.1)$  we say that a CMK instance  $\mathcal{I}$  is  $\varepsilon$ -*simple* if (i) every feasible subset of items  $C$  which can be packed in a single bin has value at most  $\varepsilon^{30} \cdot \text{OPT}(\mathcal{I})$ , (ii)  $m > \exp(\exp(\varepsilon^{-30}))$ , and (iii)  $\varepsilon \cdot m \in \mathbb{N}$ , where  $m$  is the number of bins and  $\text{OPT}(\mathcal{I})$  is the optimum value of  $\mathcal{I}$ .<sup>4</sup> For clarity, we first state our algorithmic result for the subclass of  $\varepsilon$ -simple CMK instances (see Section 3 for more details).

► **Theorem 1.** *For every  $\varepsilon \in (0, 0.1)$  and an  $\varepsilon$ -simple CMK instance  $\mathcal{I}$ , iterative randomized rounding (see Algorithm 1) returns a  $(1 - \varepsilon)$ -approximation for  $\mathcal{I}$  in time  $\left(\frac{|\mathcal{I}|}{\varepsilon}\right)^{O(1)}$ , where  $|\mathcal{I}|$  is the encoding size of  $\mathcal{I}$ .*

An in depth look into the algorithm of Fleischer et al. [12] reveals that the approximation ratio of their algorithm on  $\varepsilon$ -simple instance is not better than  $(1 - \frac{1}{e})$ , indicating the improved ratio in Theorem 1 stems from the use of the iterative approach.

<sup>2</sup> We give formal definitions of approximation schemes in Section 2.

<sup>3</sup> The paper [12] shows that the existence of an FPTAS for the single bin subproblem, as in the case of 0/1-knapsack with cardinality constraint, implies a  $(1 - 1/e)$ -approximation for the corresponding variant of SAP.

<sup>4</sup> In our discussion of  $\varepsilon$ -simple instances, we did not attempt to optimize the constants.

We give a simple reduction showing that our algorithm for  $\varepsilon$ -simple instances yields a randomized EPTAS for general CMK instances.

This essentially resolves the complexity status of CMK, since an FPTAS is ruled out [5].

► **Theorem 2.** *There is a randomized EPTAS for CMK.*

For the proof of Theorem 2 see Section 3.

### 1.3 Technical Overview

In the following, we outline our algorithmic approach and its analysis. For clarity, we focus in this section on high-level ideas and omit some technical details to improve clarity. We start with a more formal definition of CMK. An instance of CMK consists of a set of items  $I$ , a weight function  $\mathbf{w} : I \rightarrow [0, 1]$ , a value function  $\mathbf{v} : I \rightarrow \mathbb{R}_{\geq 0}$ , a number of bins  $m \in \mathbb{N}_{>0}$ , and a cardinality constraint  $k \in \mathbb{N}_{>0}$ . A *solution* is a tuple  $(C_1, \dots, C_m)$  such that for all  $j \in \{1, \dots, m\}$  it holds that  $C_j \subseteq I$ ,  $|C_j| \leq k$  and  $\mathbf{w}(C_j) = \sum_{i \in C_j} \mathbf{w}(i) \leq 1$ . The *value* of the solution  $(C_1, \dots, C_m)$  is  $\sum_{i \in S} \mathbf{v}(i)$  where  $S = \bigcup_{j=1}^m C_j$ . The goal is to find a solution of maximum value. Note that we allow the sets  $C_1, \dots, C_m$  to intersect, but if an item appears in multiple sets its value is counted only once.

#### The Algorithm

Our algorithm applies an iterative randomized rounding approach based on a configuration-LP. The use of such linear program dates back to the work of Karmarkar and Karp on bin packing [15], and such linear programs are commonly used in approximation algorithms for resource allocation problems (e.g., [3, 1, 12, 13, 17, 7]).

We use a *configuration polytope*  $P(\ell) \subseteq [0, 1]^I$ , where  $\bar{y} \in P(\ell)$  can be intuitively interpreted as “there is a way to fractionally pack the items into  $\ell$  bins such that each item  $i \in I$  is packed  $\bar{y}_i$  times”. The algorithm takes as an input a value  $\varepsilon > 0$  which serves as a discretization factor and determines the approximation ratio. Our iterative approach uses  $\frac{1}{\varepsilon}$  iterations, and each iteration packs  $\varepsilon \cdot m$  of the remaining bins. Therefore, at the beginning of the  $j$ -th iterations,  $(j-1) \cdot \varepsilon \cdot m$  bins were packed (in previous iterations) and  $(1 - (j-1) \cdot \varepsilon) \cdot m$  bins are still empty. We use  $S_j$  to denote the set of items that were not packed by the end of the  $j$ -th iteration (and thus are still available for packing). The main steps of the algorithm are as follows.

1. Initialize  $S_0 \leftarrow I$  to be all the items.
2. For  $j$  from 1 to  $\frac{1}{\varepsilon}$  do:
  - a. Solve the linear program

$$\begin{aligned}
 \max \quad & \sum_{i \in I} \bar{y}_i \cdot \mathbf{v}(i) \\
 \text{s.t.} \quad & \bar{y} \in P(m \cdot (1 - (j-1) \cdot \varepsilon)) \\
 & \bar{y}_i = 0 \quad \forall i \in I \setminus S_{j-1}
 \end{aligned} \tag{1}$$

That is, we want to obtain a maximum value using  $m \cdot (1 - (j-1) \cdot \varepsilon)$  bins and only items in  $S_{j-1}$ . Let  $\bar{y}^j$  be the solution found.

- b. Sample  $\varepsilon \cdot m$  bins according to the solution  $\bar{y}^j$  (defined more formally in later); update  $S_j$  to be  $S_{j-1}$  minus all the items packed in the current iteration.
3. Return the collection of  $m$  packed bins.

The linear program in (1) can be approximated efficiently, but cannot be solved exactly in polynomial time. For the purpose of this technical overview, we assume it can be solved exactly. In Item 2b we use the randomized rounding technique for configuration LPs of Fleischer et al. [12]. The same randomized rounding technique is commonly used by other algorithms (e.g., [3, 1, 17, 2]). We give the full details on the sampling process in Section 3.

### High Level Analysis

Let  $Q_j$  be the set of items packed in the  $j$ -th iteration (that is,  $Q_j = S_j \setminus S_{j-1}$ ). In the  $j$ -th iteration, the linear program uses  $m \cdot (1 - (j-1) \cdot \varepsilon)$  bins and attains value of  $\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)$ , with an average value of  $\frac{\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)}{m \cdot (1 - (j-1) \cdot \varepsilon)}$  per bin. As the number of bins sampled in each iteration is *small*, it can be shown that with high probability the average value per bin in the packing generated by the randomized rounding is roughly the same as the average value in the fractional solution. That is,

$$\frac{\sum_{i \in Q_j} \mathbf{v}(i)}{\varepsilon \cdot m} \approx \frac{\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)}{m \cdot (1 - (j-1) \cdot \varepsilon)},$$

or equivalently,

$$\sum_{i \in Q_j} \mathbf{v}(i) \approx \varepsilon \cdot \frac{\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)}{1 - (j-1) \cdot \varepsilon}.$$

Observe the left hand term is the value attained from items packed in the  $j$ -th iteration. In each iteration of the algorithm the distribution by which the bins are sampled is updated, so the algorithm does not pack items already packed in previous iterations (by the constraints  $\bar{y}_i = 0$  for  $i \in I \setminus S_{j-1}$  in (1)). Thus, we have that  $Q_1, \dots, Q_{\varepsilon-1}$  are disjoint. It follows that the value of the solution returned by the algorithm is

$$\mathbf{v}(I \setminus S_{\varepsilon-1}) = \sum_{j=1}^{\varepsilon-1} \sum_{i \in Q_j} \mathbf{v}(i) \approx \varepsilon \cdot \sum_{j=1}^{\varepsilon-1} \frac{\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)}{1 - (j-1) \cdot \varepsilon} \quad (2)$$

Ideally, we would like the average value per bin to be (at least)  $\frac{\text{OPT}}{m}$  in each of the solutions  $\bar{y}^j$ , where OPT is the value of the optimal solution of the instance. That is, the average value per bin in each of the iterations remains the average value per bin in the optimum. As  $\bar{y}^j$  conceptually uses  $m \cdot (1 - (j-1)\varepsilon)$  bins, this implies that

$$\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i) \gtrsim \frac{\text{OPT}}{m} \cdot m \cdot (1 - (j-1) \cdot \varepsilon) = \text{OPT} \cdot (1 - (j-1) \cdot \varepsilon), \quad (3)$$

for every  $j \in [\varepsilon-1]$ . If we assume (3) holds and plug it into (2), we get that the value of the solution returned by the algorithm is

$$\mathbf{v}(I \setminus S_{\varepsilon-1}) \approx \varepsilon \cdot \sum_{j=1}^{\varepsilon-1} \frac{\sum_{i \in I} \bar{y}_i^j \cdot \mathbf{v}(i)}{1 - (j-1) \cdot \varepsilon} \gtrsim \varepsilon \cdot \sum_{j=1}^{\varepsilon-1} \frac{\text{OPT} \cdot (1 - (j-1) \cdot \varepsilon)}{1 - (j-1) \cdot \varepsilon} = \text{OPT}.$$

That is, the algorithm returns a solution of value close to OPT (not strictly better naturally), assuming (3) holds. This leaves us with the goal of showing that (3) holds with high probability.

### Linear Structures and Equation (3)

To show that (3) holds we define a random vector  $\bar{\gamma}^j \in [0, 1]^I$  for every  $j \in [\varepsilon^{-1}]$ . We use  $\bar{\gamma}^j$  to lower bound the value of the configuration-LP. We show that with high probability (i) the value of  $\bar{\gamma}^j$  (that is,  $\sum_{i \in I} \bar{\gamma}_i^j \cdot \mathbf{v}(i)$ ) is  $\approx (1 - (j - 1)\varepsilon) \cdot \text{OPT}$  and (ii)  $\bar{\gamma}^j \in P((1 + \delta) \cdot m_j)$  where  $m_j = (1 - (j - 1) \cdot \varepsilon) \cdot m$  is the number of remaining bins at the beginning of the  $j$ -th iteration and  $\delta > 0$  is small. Once properties (i) and (ii) are shown, it follows that  $\frac{\bar{\gamma}^j}{1 + \delta}$  is a solution of high value for the linear program in the  $j$ -th iteration, and (3) immediately follows as the algorithm finds an optimal solution in every iteration. Property (i) is shown using a simple calculation of the expected value of the vector  $\bar{\gamma}^j$  followed by an application of a concentration bound which shows that with high probability the value of  $\bar{\gamma}^j$  does not deviate far from its expected value. Showing property (ii) is more challenging.

The polytope  $P(\ell)$  can be represented via a finite set of linear constraints  $\mathcal{S} \subseteq \mathbb{R}_{\geq 0}^I$  by  $P(\ell) = \{\bar{y} \in [0, 1]^I \mid \forall \bar{u} \in \mathcal{S} : \bar{u} \cdot \bar{y} \leq \ell\}$  (the set  $\mathcal{S}$  is the same for every  $\ell$ ). While  $\mathcal{S}$  is finite, its size is non-polynomial in the input instance. A naive approach to show that  $\bar{\gamma}^j \in P((1 + \delta)m_j)$  is to consider each constraint  $\bar{u} \in \mathcal{S}$  separately, and apply concentration bounds to show  $\bar{y} \cdot \bar{u} \lesssim m_j$  with high probability. Subsequently, the union bound can be used to lower bound the probability that  $\bar{y} \cdot \bar{u} \lesssim m_j$  for every  $\bar{u} \in \mathcal{S}$  simultaneously. However, due to the large number of vectors in  $\mathcal{S}$ , a direct application of the union bound does not lead to such useful lower bound.

We use a *linear structure* to overcome the above challenge. The linear structure provides an approximate representation of the configuration polytope using a small number of constraints (that is, the number of constraints only depends on  $\varepsilon$ ). As the number of constraints is reduced, we can now apply the above logic successfully – use a concentration bound to show that each constraint of the linear structure holds independently with high probability, and then use the union bound to show that all the constraints hold simultaneously with high probability. By the properties of the linear structure, once we show that all constraints hold, we are guaranteed that  $\bar{\gamma}^j \in P((1 + \delta) \cdot m_j)$ , as stated in (ii).

The concept of linear structure was introduced in [17]. It is essentially a non-constructive version of the *subset oblivious* algorithms used by the Round&Approx framework of [3]. We construct the linear structure for CMK based on ideas from [17, 1]. The structure leverages the relatively simple structure of the cardinality constraint.

### Technical Contribution

In this paper, we present the first use of an iterative randomized rounding approach of a configuration-LP for a maximization problem. As such, the paper provides the basic foundations required for the analysis of iterative randomized rounding for maximization problems. Iterative randomized rounding of a configuration-LP has been recently used for bin packing problems in [17]. Indeed, in some places the analysis only requires simple adaptations of ideas from [17]. In other parts, the adaptation is more challenging.

These challenges arise mainly due to the fact that while in bin packing all the remaining items must be fully packed by the configuration-LP, in maximization problems the remaining items may be partially selected or not selected at all by the configuration-LP. Thus, the probability of an item to be packed after  $j$  iterations may take different values for different items. In contrast, this probability is the same for all items in the case of bin packing. Similarly, while in the case of bin packing all items must be packed by the configuration-LP in every iteration, in maximization problem there is a degree of freedom in the selection of items to be packed. This, in turn, led to a different approach for the use of the linear structure.



We note that while the paper [7] deals with a generalization of CMK and uses several similar concepts (configuration LP, sampling, subset oblivious algorithms), the algorithm in [7] does not use an iterative approach. It relies on two separate stages: the first uses a randomized rounding of a configuration-LP that is solved once, and the second stage uses a combinatorial algorithm. We believe the analysis of the iterative randomized rounding algorithm presented in this paper will be useful in showing iterative randomized rounding yields an improved approximation for the Uniform 2-dimensional Vector Multiple Knapsack (2d-UMK) problem considered in [7]. The main challenge in applying our analysis to 2d-UMK is that our analysis relies on a robust linear structure, which is unlikely to exist for 2d-UMK (as that would lead to a PTAS, contradicting the hardness results in [7]). This can potentially be bypassed with the use of an analog of the linear structure that holds for 2d-UMK and adaptation of the analysis to this potential structure.

## 1.4 Organization

In Section 2 we give some definitions and notation. Section 3 presents our main algorithm and an outline of its analysis. In Section 4 we give the detailed analysis (proofs of Lemmas 6, 7, 10 and 11). The proofs of Lemma 9 and Lemma 4 are given in the full version of the paper [9].

## 2 Preliminaries

We start with some definitions and notation. Let  $\text{OPT}(I)$  be the value of an optimal solution for an instance  $I$  of a maximization problem  $\Pi$ . For  $\alpha \in (0, 1]$ , a solution  $x$  for the instance  $I$  is an  $\alpha$ -approximate solution if its value is at least  $\alpha \cdot \text{OPT}(I)$ . For  $\alpha \in (0, 1]$ , we say that  $\mathcal{A}$  is an  $\alpha$ -approximation algorithm for  $\Pi$  if for any instance  $I$  of  $\Pi$ ,  $\mathcal{A}$  outputs an  $\alpha$ -approximate solution for  $I$ . An algorithm  $\mathcal{A}$  is a *randomized*  $\alpha$ -approximation for  $\Pi$  if for any instance  $I$  of  $\Pi$  it always returns a solution for  $I$ , and the solution is an  $\alpha$ -approximate solution with probability at least  $\frac{1}{2}$ . A *polynomial-time approximation scheme* (PTAS) for a maximization problem  $\Pi$  is a family of algorithms  $(\mathcal{A}_\varepsilon)_{\varepsilon > 0}$  such that for any  $\varepsilon > 0$ ,  $\mathcal{A}_\varepsilon$  is a polynomial-time  $(1 - \varepsilon)$ -approximation algorithm for  $\Pi$ . As the running time of a PTAS may be impractically high, two restrictive classes of PTAS have been proposed in the literature:  $(\mathcal{A}_\varepsilon)_{\varepsilon > 0}$  is an *efficient* PTAS (EPTAS) if the running time of  $\mathcal{A}_\varepsilon$  is of the form  $f\left(\frac{1}{\varepsilon}\right) \cdot n^{O(1)}$ , where  $f$  is an arbitrary function, and  $n$  is the bit-length encoding size of the input instance;  $(\mathcal{A}_\varepsilon)_{\varepsilon > 0}$  is a *fully* PTAS (FPTAS) if the running time of  $\mathcal{A}_\varepsilon$  is bounded by  $\left(\frac{n}{\varepsilon}\right)^{O(1)}$ . Given a boolean expression  $\mathcal{D}$ , we define  $\mathbb{1}_{\mathcal{D}} \in \{0, 1\}$  such that  $\mathbb{1}_{\mathcal{D}} = 1$  if  $\mathcal{D}$  is true and  $\mathbb{1}_{\mathcal{D}} = 0$  otherwise.

We give an alternative definition of our problem that will be used in the technical sections. An instance of CMK is a tuple  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$ , where  $I$  is a set of items,  $\mathbf{w} : I \rightarrow [0, 1]$  is the weight function,  $\mathbf{v} : I \rightarrow \mathbb{R}_{\geq 0}$  is the value function,  $m \in \mathbb{N}_{>0}$  is the number of bins, and  $k \in \mathbb{N}_{>0}$  is the cardinality constraint. A *configuration* of the instance  $\mathcal{I}$  is  $C \subseteq I$  such that  $|C| \leq k$  and  $\mathbf{w}(C) = \sum_{i \in C} \mathbf{w}(i) \leq 1$ . Let  $\mathcal{C}_{\mathcal{I}}$  be the set of all configurations of  $\mathcal{I}$ , and  $\mathcal{C}_{\mathcal{I}}(i) = \{C \in \mathcal{C} \mid i \in C\}$  the set of all configurations which contain  $i \in I$ . When clear from the context, we simply use  $\mathcal{C} = \mathcal{C}_{\mathcal{I}}$  and  $\mathcal{C}(i) = \mathcal{C}_{\mathcal{I}}(i)$ .

A *solution* of  $\mathcal{I}$  is a tuple of  $m$  configurations  $S = (C_1, \dots, C_m) \in \mathcal{C}^m$ . The value of the solution  $S = (C_1, \dots, C_m)$  is  $\mathbf{v}(S) = \mathbf{v}\left(\bigcup_{b \in [m]} C_b\right)$  (generally, for any set  $B \subseteq A$  and a function  $f : A \rightarrow \mathbb{R}_{\geq 0}$ , we use  $f(B) = \sum_{b \in B} f(b)$ ). The objective is to find a solution of maximum value. Let  $\text{OPT}(\mathcal{I})$  be the optimal solution value for the instance  $\mathcal{I}$ , and  $|\mathcal{I}|$  the encoding size of  $\mathcal{I}$ . W.l.o.g., we consider a tuple with fewer than  $m$  configurations to be a solution. In this case, for some  $r \leq m$ , the tuple  $(C_1, \dots, C_r) \in \mathcal{C}^r$  is equivalent to the solution  $(C_1, \dots, C_r, \emptyset, \dots, \emptyset) \in \mathcal{C}^m$ .

Our main algorithm, given in Section 3, is applied to a restricted subclass of *simple* instances. We now give a more formal definition for this subclass of instances.

► **Definition 3.** Let  $\varepsilon \in (0, 0.1)$ , We say that a CMK instance  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  is  $\varepsilon$ -simple if the following conditions hold.

- For every  $C \in \mathcal{C}$ , we have that  $\mathbf{v}(C) \leq \varepsilon^{30} \cdot \text{OPT}(\mathcal{I})$ .
- $m > \exp(\exp(\varepsilon^{-30}))$
- $\varepsilon \cdot m \in \mathbb{N}$ .

We give a reduction showing that our algorithm for  $\varepsilon$ -simple instances yields a randomized EPTAS for general CMK instances.<sup>5</sup> This is formalized in the next lemma (we give the proof in [9]).

► **Lemma 4.** Given  $\varepsilon \in (0, 0.1)$  such that  $\varepsilon^{-\frac{1}{2}} \in \mathbb{N}$ , let  $\mathcal{A}$  be a randomized algorithm which returns a  $(1 - \varepsilon)$ -approximate solution for any  $\varepsilon$ -simple CMK instance  $\mathcal{I}$  in time  $\left(\frac{|\mathcal{I}|}{\varepsilon}\right)^{O(1)}$ . Then, there is a randomized EPTAS for CMK.

Theorem 2 follows from Theorem 1 and Lemma 4.

### 3 The Algorithm

In this section, we formally present our iterative randomized rounding algorithm for  $\varepsilon$ -simple CMK instances. The algorithm relies on a linear programming (LP) relaxation of CMK that we formalize through the notion of fractional solutions.

A *fractional solution* for an instance  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  is a vector  $\bar{x} \in \mathbb{R}_{\geq 0}^{\mathcal{C}}$ ; the value  $\bar{x}_C$  represents a fractional selection of the configuration  $C$  for the solution. The *coverage* of  $\bar{x}$  is the vector  $\text{cover}(\bar{x}) \in \mathbb{R}_{\geq 0}^I$  defined by

$$\forall i \in I: \quad \text{cover}_i(\bar{x}) = (\text{cover}(\bar{x}))_i = \sum_{C \in \mathcal{C}(i)} \bar{x}_C.$$

The vector  $\bar{x}$  is *feasible* if  $\text{cover}(\bar{x}) \in [0, 1]^I$ . The size of  $\bar{x}$  is  $\|\bar{x}\| = \sum_{C \in \mathcal{C}} \bar{x}_C$  (throughout this paper, for every vector  $\bar{z} \in \mathbb{R}^n$  we use  $\|\bar{z}\| = \sum_{i=1}^n |\bar{z}_i|$ ). The *value* of  $\bar{y} \in [0, 1]^I$  is  $\mathbf{v}(\bar{y}) = \sum_{i \in I} \bar{y}_i \cdot \mathbf{v}(i)$ . The *value* of  $\bar{x}$  is the value of the cover of  $\bar{x}$ , that is,  $\mathbf{v}(\bar{x}) = \mathbf{v}(\text{cover}(\bar{x}))$ . For  $\ell \in \mathbb{N}_{>0}$ , let  $[\ell] = \{1, \dots, \ell\}$ .

A solution  $S = (C_1, \dots, C_m)$  for  $\mathcal{I}$ , where  $C_1, \dots, C_m$  are disjoint and non-empty, can be encoded as a feasible fractional solution  $\bar{x} \in \{0, 1\}^{\mathcal{C}}$  defined by  $\bar{x}_{C_b} = 1$  for every  $b \in [m]$ , and  $\bar{x}_C = 0$  for every other configuration. It is easy to verify that  $\|\bar{x}\| = m$ ,  $\text{cover}_i(\bar{x}) = 1$  for every  $i \in S$ ,  $\text{cover}_i(\bar{x}) = 0$  for every  $i \in I \setminus S$ , and  $\mathbf{v}(\bar{x}) = \mathbf{v}(S)$ .

We use fractional solutions to define a linear program (LP). Let  $K$  be a set and  $\bar{\gamma} \in \mathbb{R}^K$ . The *support* of  $\bar{\gamma}$  is  $\text{supp}(\bar{\gamma}) = \{i \in K \mid \bar{\gamma}_i \neq 0\}$ . Let  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  be a CMK instance. For every set  $S \subseteq I$  of remaining items and  $\ell \in \mathbb{N}$  remaining bins, we define the configuration LP of  $S$  and  $\ell$  by

$$\begin{aligned} \text{LP}(S, \ell): \quad & \max && \mathbf{v}(\bar{x}) \\ & \text{s.t.} && \bar{x} \text{ is a feasible fractional solution for } \mathcal{I} \\ & && \text{supp}(\bar{x}) \subseteq 2^S \\ & && \|\bar{x}\| = \ell \end{aligned}$$

<sup>5</sup> In our discussion of  $\varepsilon$ -simple instances, we did not attempt to optimize the constants.

That is, in  $\text{LP}(S, \ell)$  exactly  $\ell$  configurations are selected<sup>6</sup>, and these configurations contain only items in  $S$ . We can formally define the configuration polytope  $P(\ell)$  discussed in Section 1.3 via fractional solutions by

$$P(\ell) = \{\text{cover}(\bar{x}) \mid \bar{x} \text{ is a feasible fractional solution for } \mathcal{I} \text{ and } \|\bar{x}\| \leq \ell\}. \quad (4)$$

It can be shown that  $\text{LP}(S_j, (1 - (j - 1) \cdot \varepsilon) \cdot m)$  is equivalent to the linear program in (1).

A generalization of  $\text{LP}(S, \ell)$  for the *separable assignment problem* (SAP) was considered in [12]. Given  $p_i \geq 0$  for every  $i \in I$ , the paper [12] shows that linear programs such as  $\text{LP}(S, \ell)$  admit an FPTAS whenever the *single bin problem* – of finding  $C \in \mathcal{C}$  such that  $\sum_{i \in I} p_i$  is maximized – admits an FPTAS. As the single bin case of CMK has an FPTAS (e.g., [4, 18, 10]), we get the following.

► **Lemma 5.** *There is an algorithm which given a CMK instance  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$ ,  $S \subseteq I$ ,  $\ell \in \mathbb{N}$  and  $\varepsilon > 0$ , finds a  $(1 - \varepsilon)$ -approximate solution for  $\text{LP}(S, \ell)$  in time  $\left(\frac{|I|}{\varepsilon}\right)^{O(1)}$ .*

Given a fractional solution  $\bar{x}$  such that  $\|\bar{x}\| \neq 0$ , we say that a random configuration  $R \in \mathcal{C}$  is *distributed by  $\bar{x}$* , and write  $R \sim \bar{x}$ , if  $\Pr(R = C) = \frac{\bar{x}_C}{\|\bar{x}\|}$  for all  $C \in \mathcal{C}$ .

The pseudocode of our algorithm for CMK is given in Algorithm 1. In each iteration  $1 \leq j \leq \varepsilon^{-1}$ , the algorithm uses the solution  $\bar{x}^j$  for  $\text{LP}(S_{j-1}, m_j)$  to sample  $\varepsilon \cdot m$  configurations, where  $S_{j-1}$  is the set of items remaining after iteration  $(j - 1)$ , and  $m_j$  is the number of remaining (unassigned) bins.

■ **Algorithm 1** Iterative Randomized Rounding.

---

**input** : Error parameter  $\varepsilon \in (0, 0.1)$ ,  $\varepsilon^{-\frac{1}{2}} \in \mathbb{N}$ , and an  $\varepsilon$ -simple CMK instance  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$

**output** : A solution for the instance

- 1 Initialize  $S_0 \leftarrow I$
- 2 **for**  $j = 1, \dots, \varepsilon^{-1}$  **do**
- 3     Find a  $(1 - \varepsilon)$ -approximate solution  $\bar{x}^j$  for  $\text{LP}(S_{j-1}, m_j)$ , where  $m_j = m(1 - (j - 1) \cdot \varepsilon)$ .
- 4     Sample independently  $q = \varepsilon \cdot m$  configurations  $R_1^j, \dots, R_q^j \sim \bar{x}^j$ .
- 5     Update  $S_j = S_{j-1} \setminus \left(\bigcup_{b=1}^q R_b^j\right)$ .
- 6 Return as solution  $\left(R_b^j\right)_{1 \leq j \leq \varepsilon^{-1}, 1 \leq b \leq q}$

---

Consider the execution of Algorithm 1 with the input  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  and  $\varepsilon \in (0, 0.1)$  such that  $\varepsilon^{-\frac{1}{2}} \in \mathbb{N}$ . The notations we use below, such as  $\bar{x}^j$ ,  $S_j$ , and  $R_b^j$ , refer to the variables used throughout the execution of the algorithm. Clearly, Algorithm 1 returns a solution for  $\mathcal{I}$ . Furthermore, by Lemma 5, the running time of the algorithm is polynomial in  $\mathcal{I}$  and  $\varepsilon^{-1}$ . Let  $V = \mathbf{v} \left(\bigcup_{j=1}^{\varepsilon^{-1}} \bigcup_{b=1}^q R_b^j\right) = \mathbf{v}(I \setminus S_{\varepsilon^{-1}})$  be the value of the returned solution.

## Main Lemmas

In the following, we describe the main lemmas we prove in order to lower bound the value of  $V$ . The proofs of the lemmas are given in Section 4 and the full version of the paper [9].

<sup>6</sup> Note that  $\bar{x}_0$  may be greater than 1.

## 27:10 An EPTAS for Cardinality Constrained Multiple Knapsack

A simple calculation shows that the expected value of  $\mathbf{v}(R_b^j)$ , given all the samples up to (and including) iteration  $(j-1)$ , is  $\frac{\mathbf{v}(\bar{x}^j)}{m \cdot (1 - (j-1)\varepsilon)}$ . To compute the expected value of  $\mathbf{v}\left(\bigcup_{b=1}^q R_b^j\right)$ , we need to take into consideration events in which an item  $i \in I$  appears in several configurations among  $R_1^j, \dots, R_q^j$ . In Section 4.2 we show that, since only a small number of configurations are sampled in each iteration (in comparison to the overall remaining number of bins), such events have small effect on the expected value (with the exception of the last  $\varepsilon^{-\frac{1}{2}}$  iterations). This observation is coupled with a concentration bound to prove the next lemma.

► **Lemma 6.** *With probability at least  $1 - \exp(-\varepsilon^{-8})$ , it holds that*

$$V = \mathbf{v}(I \setminus S_{\varepsilon^{-1}}) \geq \sum_{j=1}^{\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}} \mathbf{v}(\bar{x}^j) \cdot \frac{(\varepsilon - \varepsilon^{\frac{3}{2}})}{1 - (j-1)\varepsilon} - \varepsilon^9 \cdot \text{OPT}(\mathcal{I}).$$

Lemma 6 is the formal statement of (2). Lemma 6 essentially reduces the problem of deriving a lower bound for  $V$  to obtaining a lower bound on  $\mathbf{v}(\bar{x}^j)$ .

To obtain a lower bound for  $\mathbf{v}(\bar{x}^j)$  we use the following steps. We define random vectors  $\bar{\gamma}^j \in [0, 1]^I$  for every  $j \in [\varepsilon^{-1}]$  such that  $\mathbf{v}(\bar{\gamma}^j)$  is high, and there is  $\bar{z}^j$  such that  $\text{cover}(\bar{z}^j) = \bar{\gamma}^j$  and  $\|\bar{z}^j\| \approx m_j$ . We scale down  $\bar{z}^j$  to obtain a solution for  $\text{LP}(S_{j-1}, m_j)$  of value  $\approx \mathbf{v}(\bar{\gamma}^{j-1})$ , and consequently get a lower bound for  $\mathbf{v}(\bar{x}^j)$ . We use a *linear structure* defined below, to show the existence of  $\bar{z}^j$ . We further use auxiliary random vectors  $\bar{\lambda}^j$  to define  $\bar{\gamma}^j$ .

Let  $(\Omega, \mathcal{F}, \text{Pr})$  be the probability space defined by the execution of the algorithm. Define the  $\sigma$ -algebras  $\mathcal{F}_0 = \{\emptyset, \Omega\}$  and  $\mathcal{F}_j = \sigma(\{R_b^{j'} \mid 1 \leq j' \leq j, 1 \leq b \leq q\})$ . That is,  $\mathcal{F}_j$  describes events which only depend on the outcomes of the random sampling up to (and including) the  $j$ -th iteration of the algorithm. We follow the standard definition of conditional probabilities and expectations given  $\sigma$ -algebras (see, e.g., [6]).

Fix an optimal solution  $(C_1^*, \dots, C_m^*)$  for the instance and let  $S^* = \bigcup_{j=1}^m C_j^*$  be the set of items in this solution. Also, given a set  $S \subseteq I$  denote by  $\mathbb{1}_S$  the vector  $\bar{z} \in \{0, 1\}^I$  satisfying  $\bar{z}_i = 1$  for  $i \in S$ , and  $\bar{z}_i = 0$  otherwise.

We define  $\bar{\gamma}^j$  and  $\bar{\lambda}^j$  inductively using  $S^*$ . Define  $\bar{\gamma}^0 = \mathbb{1}_{S^*}$ , that is  $\bar{\gamma}_i^0 = 1$  for every  $i \in S^*$  and  $\bar{\gamma}_i^0 = 0$  for every  $i \in I \setminus S^*$ . For every  $j \in [\varepsilon^{-1} - 1]$  define  $\bar{\lambda}^j \in \mathbb{R}_{\geq 0}^I$  by

$$\bar{\lambda}_i^j = \frac{1 - j \cdot \varepsilon}{1 - (j-1)\varepsilon} \cdot \frac{1}{\text{Pr}(i \in S_j \mid \mathcal{F}_{j-1})} \cdot \bar{\gamma}_i^{j-1} \quad (5)$$

for all  $i \in S_{j-1}$  and  $\bar{\lambda}_i^j = 0$  for  $i \notin S_{j-1}$ . Intuitively, the expression  $\text{Pr}(i \in S_j \mid \mathcal{F}_{j-1})$  in (5) is the probability that item  $i$  will still be available for packing after the  $j$ -th iteration, where the probability is calculated at the end of the iteration  $j-1$ . Also, for every  $j \in [\varepsilon^{-1} - 1]$  define  $\bar{\gamma}^j \in \mathbb{R}_{\geq 0}^I$  by

$$\bar{\gamma}_i^j = \mathbb{1}_{i \in S_j} \cdot \bar{\lambda}_i^j \quad \forall i \in I. \quad (6)$$

Observe that  $\bar{\lambda}^j$  is  $\mathcal{F}_{j-1}$ -measurable random variable whereas  $\bar{\gamma}^j$  is  $\mathcal{F}_j$ -measurable. Intuitively, this means that the value of  $\bar{\lambda}^j$  is known by the end of the  $(j-1)$ -th iteration, while the value of  $\bar{\gamma}^j$  is only known by the end of the  $j$ -th iteration.

The lower bound on  $\mathbf{v}(\bar{\gamma}^{j-1})$  relies on a simple calculation of expectations followed by a concentration bound. By induction it can be shown that  $\mathbb{E}[\bar{\gamma}_i^{j-1}] = (1 - (j-1)\varepsilon) \cdot \mathbb{1}_{i \in S^*}$ , and therefore,

$$\mathbb{E}[\mathbf{v}(\bar{\gamma}^{j-1})] = (1 - (j-1)\varepsilon) \cdot \mathbf{v}(S^*) = (1 - (j-1)\varepsilon) \cdot \text{OPT}(\mathcal{I}).$$

We use concentration bounds to show that indeed  $\mathbf{v}(\bar{\gamma}^{j-1})$  does not deviate from its expected value.

► **Lemma 7.** *With probability at least  $1 - \exp(-\varepsilon^{-20})$ , it holds that*

$$\forall j \in [\varepsilon^{-1}] : \quad \mathbf{v}(\bar{\gamma}^{j-1}) \geq (1 - \varepsilon(j-1)) \cdot \text{OPT}(\mathcal{I}) - \varepsilon^3 \cdot \text{OPT}(\mathcal{I}).$$

We give the proof of Lemma 7 in the full version of the paper [9].

Our next challenge is to show that there is a solution for  $\text{LP}(S_{j-1}, m_j)$  whose cover is roughly  $\bar{\gamma}^{j-1}$ , which can be alternatively stated as  $\bar{\gamma}^{j-1} \in P(\ell)$  where  $\ell \approx m_j$ , and  $P(\ell)$  is as defined in (4). To this end, we introduce a *linear structure* for CMK. The main idea in linear structures is that they allow us to determine that  $\bar{\gamma}^j \in P(\ell)$  by checking if  $\bar{\gamma}^j$  satisfies a small number of linear inequalities.

Given a vector  $\bar{u} \in \mathbb{R}_{\geq 0}^I$  which defines an inequality in the linear structure, we use concentration bounds to show that  $\bar{\gamma}^j \cdot \bar{u} \leq \mathbb{E}[\bar{\gamma}^j \cdot \bar{u}] + \xi$ , where  $\xi$  is an error term. The concentration bounds we use only provide useful guarantees if the error term  $\xi$  is of order of the maximum sum of entries in  $\bar{u}$  w.r.t. a single configuration, that is,  $\text{tol}(\bar{u}) = \max \{ \sum_{i \in C} \bar{u}_i \mid C \in \mathcal{C} \}$ . We refer to the value  $\text{tol}(\bar{u})$  as the *tolerance* of  $\bar{u}$ . We consequently require the linear structure to be robust to additive errors of order of the tolerance. Also, we say that  $S \subseteq I$  can be *packed into  $\ell \in \mathbb{N}$  bins* if there are  $\ell$  configurations  $C_1, \dots, C_\ell \in \mathcal{C}$  such that  $\bigcup_{b=1}^{\ell} C_b = S$ .

► **Definition 8 (Linear Structure).** *Let  $(I, \mathbf{w}, \mathbf{v}, m, k)$  be a CMK instance and  $\delta > 0$  a parameter. Also, consider a subset  $S \subseteq I$  such that  $S$  can be packed in  $\ell \in \mathbb{N}$  bins. A  $\delta$ -linear structure of  $S$  is a set of vectors  $\mathcal{L} \subseteq \mathbb{R}_{\geq 0}^I$  which satisfy the following property.*

■ *Let  $\bar{y} \in ([0, 1] \cap \mathbb{Q})^I$ ,  $0 < \alpha < 1$  and  $t > 0$ , such that*

1.  $\text{supp}(\bar{y}) \subseteq S$
2.  $\forall \bar{u} \in \mathcal{L} : \quad \bar{u} \cdot \bar{y} \leq \alpha \cdot \bar{u} \cdot \mathbb{1}_S + t \cdot \text{tol}(\bar{u})$

*Then, there is a fractional solution  $\bar{x}$  whose cover is  $\bar{y}$  and  $\|\bar{x}\| \leq \alpha \cdot \ell + 20\delta\ell + (t+1) \cdot \exp(\delta^{-5})$ .*

The size of the structure  $\mathcal{L}$  is  $|\mathcal{L}|$ .

Alternatively, a  $\delta$ -linear structure guarantees for  $S$  that for every  $\bar{y} \in [0, 1]^I$  with rational entries,  $0 < \alpha < 1$  and  $t > 0$ , if  $\text{supp}(\bar{y}) \subseteq S$  and  $\bar{y}$  satisfies  $|\mathcal{L}|$  linear inequalities, then  $\bar{y} \in P(\alpha \cdot \ell + 20\delta\ell + (t+1) \cdot \exp(\delta^{-5}))$ .

In [9] we prove the next result.

► **Lemma 9.** *Given  $\delta > 0$ , let  $I = (I, \mathbf{w}, \mathbf{v}, m, k)$  be a CMK instance, and  $S \subseteq I$  a subset which can be packed into  $\ell > \exp(\delta^{-5})$  bins. Then there is a  $\delta$ -linear structure  $\mathcal{L}$  of  $S$  of size at most  $\exp(\delta^{-4})$ .*

The above lemma is an adaptation of a construction of [17] used to solve the vector bin packing problem, in which there are additional requirements for the packing of  $S$ . Our adaptation leverages the relative simplicity of a cardinality constraint to omit these additional requirements.

We use Lemma 9 to show the existence of an  $\varepsilon^2$ -linear structure of  $S^*$ , where  $S^*$  is the set of items in an optimal solution. We use the linear structure to show the existence of a fractional solution  $\bar{z}^j$  such that  $\text{cover}(\bar{z}^j) = \bar{\gamma}^{j-1}$  and  $\|\bar{z}^j\| \approx (1 - (j-1)\varepsilon)m$  for every  $j \in [\varepsilon^{-1}]$ . A simple scaling is then used to construct a solution for  $\text{LP}(S_{j-1}, m_j)$  and establish the following lower bound on  $\mathbf{v}(\bar{x}^j)$ .

► **Lemma 10.** *With probability at least  $1 - \exp(-\varepsilon^{-20})$ , it holds that*

$$\forall j \in [\varepsilon^{-1}] : \quad \mathbf{v}(\bar{x}^j) \geq (1 - \varepsilon) \cdot \left( 1 - \frac{30 \cdot \varepsilon^2}{1 - (j-1)\varepsilon} \right) \cdot \mathbf{v}(\bar{\gamma}^{j-1}).$$

We give the proof of Lemma 10 in [9]. Together, Lemma 10 and Lemma 7 essentially give the formal proof of (3). Finally, using Lemmas 6, 7, and 10, we obtain the next result, whose proof is given in [9].

► **Lemma 11.** *With probability at least  $1 - \exp(-\varepsilon^{-5})$ , it holds that  $V \geq (1 - 60\sqrt{\varepsilon}) \cdot \text{OPT}(\mathcal{I})$ .*

Theorem 1 follows directly from Lemma 11.

## 4 The Analysis

Consider an execution of Algorithm 1 with the input  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  and  $\varepsilon > 0$ . We use the notation and definitions as given in Section 3. Also, let  $\bar{y}^j = \text{cover}(\bar{x}^j)$  be the coverage of  $\bar{x}^j$ . Observe  $\bar{x}^j$  and  $\bar{y}^j$  are  $\mathcal{F}_{j-1}$ -measurable. That is, their values are determined by the outcomes of the samples up to (and including) the  $j - 1$  iteration. As in Section 3 we let  $(C_1^*, \dots, C_m^*)$  be an optimal solution for the instance  $\mathcal{I}$ . We define  $S^* = \bigcup_{b=1}^m C_b^*$  and  $\text{OPT} = \mathbf{v}(S^*) = \text{OPT}(\mathcal{I})$ .

### 4.1 Concentration Bounds

Before we give the proofs of Lemmas 6, 7, 10, and 11, we need to introduce some concentration bounds for *self-bounding functions*.

► **Definition 12.** *A non-negative function  $f : \mathcal{X}^n \rightarrow \mathbb{R}_{\geq 0}$  is called self-bounding if there exist  $n$  functions  $f_1, \dots, f_n : \mathcal{X}^{n-1} \rightarrow \mathcal{R}$  such that for all  $x = (x_1, \dots, x_n) \in \mathcal{X}^n$ ,*

$$0 \leq f(x) - f_t(x^{(t)}) \leq 1, \quad \text{and}$$

$$\sum_{t=1}^n \left( f(x) - f_t(x^{(t)}) \right) \leq f(x),$$

where  $x^{(t)} = (x_1, \dots, x_{t-1}, x_{t+1}, \dots, x_n) \in \mathcal{X}^{n-1}$  is obtained by dropping the  $t$ -th component of  $x$ .

We rely on the following concentration bound due to Boucheron, Lugosi and Massart [3].

► **Lemma 13.** *Let  $f : \mathcal{X}^n \rightarrow \mathbb{R}_{\geq 0}$  be a self-bounding function and let  $X_1, \dots, X_n \in \mathcal{X}$  be independent random variables. Define  $Z = f(X_1, \dots, X_n)$ . Then the following holds:*

1.  $\Pr(Z \geq \mathbb{E}[Z] + t) \leq \exp\left(-\frac{t^2}{2 \cdot \mathbb{E}[Z] + \frac{t}{3}}\right)$ , for every  $t \geq 0$ .
2.  $\Pr(Z \leq \mathbb{E}[Z] - t) \leq \exp\left(-\frac{t^2}{2 \cdot \mathbb{E}[Z]}\right)$ , for every  $t > 0$ .

The setting considered in [3] can be trivially extended to a setting in which the random variable are conditionally independent on a  $\sigma$ -algebra  $\mathcal{G}$  (see [6] for the definition of conditional independence) and the function  $f$  itself is a  $\mathcal{G}$ -measurable random function. This is formally stated in the next lemma.

► **Lemma 14.** *Let  $(\Omega, \mathcal{F}, \Pr)$  be a finite probability space and let  $\mathcal{G} \subseteq \mathcal{F}$  be a  $\sigma$ -algebra. Let  $D$  be a finite set of self-bounding function from  $\mathcal{X}^\ell$  to  $\mathbb{R}_{\geq 0}$  and let  $f \in D$  be a  $\mathcal{G}$ -measurable random function. Also, let  $X_1, \dots, X_\ell \in \mathcal{X}$  be random variables which are conditionally independent given  $\mathcal{G}$ . Define  $Z = f(X_1, \dots, X_n)$ . Then the following holds:*

1.  $\Pr(Z \geq \mathbb{E}[Z | \mathcal{G}] + t | \mathcal{G}) \leq \exp\left(-\frac{t^2}{2 \cdot \mathbb{E}[Z | \mathcal{G}] + \frac{t}{3}}\right)$ , for every  $t \geq 0$ .
2.  $\Pr(Z \leq \mathbb{E}[Z | \mathcal{G}] - t | \mathcal{G}) \leq \exp\left(-\frac{t^2}{2 \cdot \mathbb{E}[Z | \mathcal{G}]}\right)$  for every  $t \geq 0$ .

The generalization in Lemma 14 is required since the variables  $R_1^j, \dots, R_q^j$  are dependent for  $q > 1$  while being conditionally independent given the variables  $R_b^{j'}$  for every  $j' < j$  and  $b \in [q]$ . The following construction for self-bounding function was shown in [7].

► **Lemma 15.** Let  $\mathcal{I} = (I, \mathbf{w}, \mathbf{v}, m, k)$  be a CMK instance, and  $h : I \rightarrow \mathbb{R}_{\geq 0}$ . For some  $\ell \in \mathbb{N}_{>0}$  define  $f : \mathcal{C}^\ell \rightarrow \mathbb{R}_{\geq 0}$  by  $f(C_1, \dots, C_\ell) = \frac{h(\bigcup_{i \in [\ell]} C_i)}{\eta}$  where  $\eta \geq \max_{C \in \mathcal{C}} h(C)$ . Then  $f$  is self-bounding.

## 4.2 The proof of Lemma 6

The first step towards the proof of Lemma 6 is to show a lower bound on the probability of an item to appear in one of the sampled configurations  $R_1^j, \dots, R_q^j$  in terms of  $\bar{y}_i^j$ .

► **Lemma 16.** For every  $i \in I$  and  $j \in [\varepsilon^{-1}]$  it holds that  $\Pr(i \in S_{j-1} \setminus S_j \mid \mathcal{F}_{j-1}) \geq 1 - \exp\left(-\varepsilon \cdot \frac{\bar{y}_i^j}{1-(j-1)\varepsilon}\right)$ .

**Proof.** By a simple calculation,

$$\begin{aligned}
\Pr(i \in S_{j-1} \setminus S_j \mid \mathcal{F}_{j-1}) &= \Pr\left(i \in \bigcup_{b=1}^q R_b^j \mid \mathcal{F}_{j-1}\right) \\
&= 1 - \Pr\left(i \notin \bigcup_{b=1}^q R_b^j \mid \mathcal{F}_{j-1}\right) \\
&= 1 - \prod_{b=1}^q \Pr\left(i \notin R_b^j \mid \mathcal{F}_{j-1}\right) \\
&= 1 - \prod_{b=1}^q \left(1 - \Pr\left(i \in R_b^j \mid \mathcal{F}_{j-1}\right)\right) \\
&= 1 - \prod_{b=1}^q \left(1 - \Pr\left(R_b^j \in \mathcal{C}(i) \mid \mathcal{F}_{j-1}\right)\right).
\end{aligned} \tag{7}$$

The third equality holds as  $R_1^j, \dots, R_q^j$  are conditionally independent given  $\mathcal{F}_{j-1}$ . Therefore, by (7) and since the configurations are distributed by  $\bar{x}^j$  we have

$$\begin{aligned}
\Pr(i \in S_{j-1} \setminus S_j \mid \mathcal{F}_{j-1}) &= 1 - \prod_{b=1}^q \left(1 - \Pr\left(R_b^j \in \mathcal{C}(i) \mid \mathcal{F}_{j-1}\right)\right) \\
&= 1 - \left(1 - \frac{\sum_{C \in \mathcal{C}(i)} \bar{x}_C^j}{\|\bar{x}^j\|}\right)^q \\
&= 1 - \left(1 - \frac{\bar{y}_i^j}{m \cdot (1 - (j-1) \cdot \varepsilon)}\right)^{\varepsilon \cdot m} \\
&= 1 - \left(\left(1 - \frac{\bar{y}_i^j}{m \cdot (1 - (j-1) \cdot \varepsilon)}\right)^{\frac{m \cdot (1 - (j-1) \cdot \varepsilon)}{\bar{y}_i^j}}\right)^{\frac{\varepsilon \cdot \bar{y}_i^j}{(1 - (j-1) \cdot \varepsilon)}} \\
&\geq 1 - (e^{-1})^{\frac{\varepsilon \cdot \bar{y}_i^j}{(1 - (j-1) \cdot \varepsilon)}} \\
&= 1 - \exp\left(-\frac{\varepsilon \cdot \bar{y}_i^j}{(1 - (j-1) \cdot \varepsilon)}\right).
\end{aligned}$$

The inequality holds since  $(1 - \frac{1}{x})^x \leq \frac{1}{e}$  for all  $x \geq 1$ . ◀

## 27:14 An EPTAS for Cardinality Constrained Multiple Knapsack

The next lemma uses Lemma 16 to lower bound the total value of sampled configurations in the  $j$ -th iteration.

► **Lemma 17.** *For all  $j \in [\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}]$  it holds that*

$$\mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] \geq \mathbf{v}(\bar{x}^j) \cdot \left(\varepsilon - \varepsilon^{\frac{3}{2}}\right) \frac{1}{1 - (j-1)\varepsilon}.$$

**Proof.** By Lemma 16 we get

$$\begin{aligned} \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] &= \sum_{i \in I} \mathbf{v}(i) \cdot \Pr(i \in S_{j-1} \setminus S_j \mid \mathcal{F}_{j-1}) \\ &\geq \sum_{i \in I} \mathbf{v}(i) \cdot \left(1 - \exp\left(-\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon}\right)\right) \\ &\geq \sum_{i \in I} \mathbf{v}(i) \cdot \left(\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon} - \left(\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon}\right)^2\right) \tag{8} \\ &= \sum_{i \in I} \mathbf{v}(i) \cdot \left(\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon} \cdot \left(1 - \varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon}\right)\right). \end{aligned}$$

The second inequality follows from  $1 - \exp(-x) \geq x - x^2$  for all  $x \geq 0$ . By (8) we have

$$\begin{aligned} \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] &\geq \sum_{i \in I} \mathbf{v}(i) \cdot \left(\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon} \cdot \left(1 - \varepsilon \cdot \frac{1}{1 - (\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}} - 1)\varepsilon}\right)\right) \\ &= \sum_{i \in I} \mathbf{v}(i) \cdot \left(\varepsilon \cdot \frac{\bar{y}_i^j}{1 - (j-1)\varepsilon} \cdot \left(1 - \frac{\varepsilon}{\varepsilon + \varepsilon^{\frac{1}{2}}}\right)\right) \\ &= \frac{1}{1 - (j-1)\varepsilon} \cdot \left(\varepsilon - \frac{\varepsilon^2}{\varepsilon + \varepsilon^{\frac{1}{2}}}\right) \cdot \sum_{i \in I} \mathbf{v}(i) \cdot \bar{y}_i^j \\ &= \frac{1}{1 - (j-1)\varepsilon} \cdot \left(\varepsilon - \frac{1}{\varepsilon^{-1} + \varepsilon^{-\frac{3}{2}}}\right) \cdot \mathbf{v}(\bar{x}^j) \\ &\geq \mathbf{v}(\bar{x}^j) \cdot \left(\varepsilon - \varepsilon^{\frac{3}{2}}\right) \frac{1}{1 - (j-1)\varepsilon}. \end{aligned}$$

The first inequality holds since  $j \leq \varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}$  and since  $\bar{x}^j$  is a feasible solution for  $\text{LP}(S, m_j)$ ; thus,  $\bar{y}^j \in [0, 1]^I$ . ◀

We can also use Lemma 14 to show that the value of the configurations sampled in the  $j$ -th iteration does not deviate significantly from its expected value.

► **Lemma 18.** *For all  $j \in [\varepsilon^{-1}]$  it holds that*

$$\Pr\left(\mathbf{v}(S_{j-1} \setminus S_j) \leq \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I})\right) \leq \exp(-\varepsilon^{-9}).$$

**Proof.** Recall that  $q = \varepsilon \cdot m$ . Define a function  $f : \mathcal{C}^q \rightarrow \mathbb{R}_{\geq 0}$  by  $f(X) = \frac{\mathbf{v}(S)}{\varepsilon^{30} \cdot \text{OPT}(\mathcal{I})}$  for all  $X = (C_1, \dots, C_q) \in \mathcal{C}^q$ . Since  $\mathcal{I}$  is  $\varepsilon$ -simple it holds that  $\mathbf{v}(C) \leq \varepsilon^{30} \cdot \text{OPT}$  for all  $C \in \mathcal{C}$ , thus, by Lemma 15 it follows that  $f$  is a self-bounding function. Therefore,



$$\begin{aligned}
& \Pr \left( \mathbf{v}(S_{j-1} \setminus S_j) \leq \mathbb{E} [\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \mid \mathcal{F}_{j-1} \right) \\
&= \Pr \left( f(R_1^j, \dots, R_q^j) \leq \mathbb{E} \left[ f(R_1^j, \dots, R_q^j) \mid \mathcal{F}_{j-1} \right] - \varepsilon^{-20} \mid \mathcal{F}_{j-1} \right) \\
&\leq \exp \left( - \frac{\varepsilon^{-40}}{2 \cdot \mathbb{E} \left[ f(R_1^j, \dots, R_q^j) \mid \mathcal{F}_{j-1} \right]} \right).
\end{aligned} \tag{9}$$

The inequality holds by Lemma 14. In addition, since  $R_1^j, \dots, R_q^j$  is a solution for  $\mathcal{I}$ , it also holds that

$$\mathbb{E} \left[ f(R_1^j, \dots, R_q^j) \mid \mathcal{F}_{j-1} \right] \leq \frac{\text{OPT}(\mathcal{I})}{\varepsilon^{30} \cdot \text{OPT}(\mathcal{I})} = \varepsilon^{-30}. \tag{10}$$

Hence, by the above

$$\begin{aligned}
& \Pr \left( \mathbf{v}(S_{j-1} \setminus S_j) \leq \mathbb{E} [\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \mid \mathcal{F}_{j-1} \right) \\
&\leq \exp \left( - \frac{\varepsilon^{-40}}{2 \cdot \mathbb{E} \left[ f(R_1^j, \dots, R_q^j) \mid \mathcal{F}_{j-1} \right]} \right) \\
&\leq \exp \left( - \frac{\varepsilon^{-40}}{2 \cdot \varepsilon^{-30}} \right) \\
&= \exp \left( - \frac{\varepsilon^{-10}}{2} \right) \\
&\leq \exp(-\varepsilon^{-9}).
\end{aligned} \tag{11}$$

The first inequality holds by (9). The second inequality follows from (10). For the last inequality, recall that  $\varepsilon < 0.1$ . Therefore, by (11) it holds that (unconditionally on  $\mathcal{F}_{j-1}$ ),

$$\Pr \left( \mathbf{v}(S_{j-1} \setminus S_j) \leq \mathbb{E} [\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \right) \leq \exp(-\varepsilon^{-9}). \quad \blacktriangleleft$$

The proof of Lemma 6 follows from Lemma 17 and Lemma 18.

► **Lemma 6.** *With probability at least  $1 - \exp(-\varepsilon^{-8})$ , it holds that*

$$V = \mathbf{v}(I \setminus S_{\varepsilon^{-1}}) \geq \sum_{j=1}^{\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}} \mathbf{v}(\bar{x}^j) \cdot \frac{(\varepsilon - \varepsilon^{\frac{3}{2}})}{1 - (j-1)\varepsilon} - \varepsilon^9 \cdot \text{OPT}(\mathcal{I}).$$

**Proof.**

$$\begin{aligned}
 & \Pr \left( \mathbf{v}(I \setminus S_{\varepsilon^{-1}}) \geq \sum_{j=1}^{\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}} \mathbf{v}(\bar{x}^j) \cdot \frac{(\varepsilon - \varepsilon^{\frac{3}{2}})}{1 - (j-1)\varepsilon} - \varepsilon^9 \cdot \text{OPT}(\mathcal{I}) \right) \\
 & \geq \Pr \left( \bigwedge_{j \in [\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}]} \left( \mathbf{v}(S_{j-1} \setminus S_j) \geq \mathbf{v}(\bar{x}^j) \cdot \frac{(\varepsilon - \varepsilon^{\frac{3}{2}})}{1 - (j-1)\varepsilon} - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \right) \right) \\
 & \geq \Pr \left( \bigwedge_{j \in [\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}]} \left( \mathbf{v}(S_{j-1} \setminus S_j) \geq \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \right) \right) \\
 & \geq 1 - \Pr \left( \bigvee_{j \in [\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}]} \left( \mathbf{v}(S_{j-1} \setminus S_j) < \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \right) \right). \tag{12}
 \end{aligned}$$

The first inequality holds because if all  $\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}$  events in the second expression occur, then so is the event in the first expression. The second inequality holds by Lemma 17. By (12) and the union bound

$$\begin{aligned}
 & \Pr \left( \mathbf{v}(I \setminus S_{\varepsilon^{-1}}) \geq \sum_{j=1}^{\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}} \mathbf{v}(\bar{x}^j) \cdot \frac{(\varepsilon - \varepsilon^{\frac{3}{2}})}{1 - (j-1)\varepsilon} - \varepsilon^9 \cdot \text{OPT}(\mathcal{I}) \right) \\
 & \geq 1 - \sum_{j \in [\varepsilon^{-1} - \varepsilon^{-\frac{1}{2}}]} \Pr \left( \mathbf{v}(S_{j-1} \setminus S_j) < \mathbb{E}[\mathbf{v}(S_{j-1} \setminus S_j) \mid \mathcal{F}_{j-1}] - \varepsilon^{10} \cdot \text{OPT}(\mathcal{I}) \right) \\
 & \geq 1 - \varepsilon^{-1} \cdot \exp(-\varepsilon^{-9}) \\
 & \geq 1 - \exp(-\varepsilon^{-8}).
 \end{aligned}$$

The second inequality holds Lemma 18. For the last inequality, recall that  $\varepsilon < 0.1$ .  $\blacktriangleleft$

The remaining proofs are given in the full version of the paper [9].

---

## References

- 1 Nikhil Bansal, Marek Eliás, and Arindam Khan. Improved approximation for vector bin packing. In *Proc. SODA*, pages 1561–1579. SIAM, 2016.
- 2 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*, pages 13–25. SIAM, 2014.
- 3 Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. A sharp concentration inequality with applications. *Random Structures & Algorithms*, 16(3):277–292, 2000.
- 4 Alberto Caprara, Hans Kellerer, Ulrich Pferschy, and David Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *Eur. J. Oper. Res.*, 123(2):333–345, 2000.
- 5 Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

- 6 Yuan Shih Chow and Henry Teicher. *Probability theory: independence, interchangeability, martingales*. Springer Science & Business Media, 1997.
- 7 Tomer Cohen, Ariel Kulik, and Hadas Shachnai. Improved approximation for two-dimensional vector multiple knapsack. In *34th International Symposium on Algorithms and Computation, ISAAC*, pages 20:1–20:17, 2023.
- 8 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An afptas for bin packing with partition matroid via a new method for lp rounding. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.
- 9 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An EPTAS for cardinality constrained multiple knapsack via iterative randomized rounding. *arXiv preprint*, 2023. [arXiv:2308.12622](https://arxiv.org/abs/2308.12622).
- 10 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An fptas for budgeted laminar matroid independent set. *Operations Research Letters*, 51(6):632–637, 2023.
- 11 Leah Epstein and Asaf Levin. Afptas results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.
- 12 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Math. Oper. Res.*, 36(3):416–431, 2011.
- 13 Klaus Jansen. Parameterized approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 39(4):1392–1412, 2010.
- 14 Klaus Jansen. A fast approximation scheme for the multiple knapsack problem. In *Proc. SOFSEM*, pages 313–324, 2012.
- 15 Narendra Karmarkar and Richard M Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 312–320. IEEE, 1982.
- 16 Hans Kellerer. A polynomial time approximation scheme for the multiple knapsack problem. In *RANDOM-APPROX*, pages 51–62. Springer, 1999.
- 17 Ariel Kulik, Matthias Mnich, and Hadas Shachnai. Improved approximations for vector bin packing via iterative randomized rounding. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1366–1376, 2023.
- 18 Wenxin Li, Joohyun Lee, and Ness Shroff. A faster fptas for knapsack problem with cardinality constraint. *Discrete Applied Mathematics*, 315:71–85, 2022.



# Rectangle Tiling Binary Arrays

Pratik Ghosal  

Indian Institute of Technology, Palakkad, India

Syed Mohammad Meesum  

Krea University, India

Katarzyna Paluch  

University of Wrocław, Wrocław, Poland

---

## Abstract

The problem of rectangle tiling binary arrays is defined as follows. Given an  $n \times n$  array  $A$  of zeros and ones and a natural number  $p$ , our task is to partition  $A$  into at most  $p$  rectangular tiles, so that the maximal weight of a tile is minimized. A tile is any rectangular subarray of  $A$ . The weight of a tile is the sum of elements that fall within it. We present a linear ( $O(n^2)$ ) time  $(\frac{3}{2} + \frac{p^2}{w(A)})$ -approximation algorithm for this problem, where  $w(A)$  denotes the weight of the whole array  $A$ . This improves on the previously known approximation with the ratio 2 when  $\frac{p^2}{w(A)} < 1/2$ .

The result is best possible in the following sense. The algorithm employs the lower bound of  $L = \lceil \frac{w(A)}{p} \rceil$ , which is the only known and used bound on the optimum in all algorithms for rectangle tiling. We prove that a better approximation factor for the binary RTILE cannot be achieved using  $L$ , because there exist arrays, whose every partition contains a tile with weight at least  $(\frac{3}{2} + \frac{p}{w(A)})L$ . We also consider the dual problem of rectangle tiling for binary arrays, where we are given an upper bound on the weight of the tiles, and we have to cover the array  $A$  with the minimum number of non-overlapping tiles. Both problems have natural extensions to  $d$ -dimensional versions, for which we provide analogous results.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems

**Keywords and phrases** Rectangle Tiling, RTILE, DRTILE

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.28

**Category** APPROX

**Related Version** *Full Version:* <https://arxiv.org/pdf/2007.14142>

**Funding** This work was partially funded by Polish National Science Center grant UMO-2018/29/B/ST6/02633.

## 1 Introduction

In this paper we study several variants of the rectangle tiling problem. These problems belong to a very wide class of discrete optimization tiling problems. As an input, we are given a two-dimensional array  $A[1..n, 1..n]$ , where each cell  $A[i, j]$  has a non-negative weight.

**RTILE.** Given a two-dimensional array  $A$  of size  $n \times n$  and a natural number  $p$ , we partition  $A$  into at most  $p$  rectangular subarrays, called *tiles*, so that the maximum weight of any tile is minimized. In other words, we have to cover  $A$  with tiles such that no two tiles overlap, while minimizing the weight of any tile. The weight of a tile is the sum of the elements that fall within it.

**DRTILE.** A natural variant of RTILE is called the DRTILE problem. The DRTILE problem is a dual of the RTILE problem, where we are given an upper bound  $W$  on the weight of the tiles, and we have to cover the array  $A$  with the minimum number of non-overlapping tiles.



© Pratik Ghosal, Syed Mohammad Meesum, and Katarzyna Paluch;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 28; pp. 28:1–28:17



Leibniz International Proceedings in Informatics

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

These two problems have a natural extension to  $d$  dimensions. Here the input is a  $d$ -dimensional array  $A$  of size  $n$  in each dimension and we have to partition  $A$  into non-overlapping  $d$ -dimensional tiles so that the optimality criterion of the RTILE/DRTILE problem is satisfied.

In this paper we consider a special case of the RTILE/DRTILE problem, where each cell has a binary weight, i.e., the weight of any cell is either 0 or 1. We extend our approach to solve the  $d$ -dimensional binary RTILE/DRTILE problem.

**Motivation.** The RTILE/ DRTILE problem is a general problem in combinatorial optimization that has a wide variety of applications in real life. These include load balancing in parallel computing environments, video compression, data partitioning, database mining, and building equisum histogram on two or more attributes. A detailed description of the practical applications of RTILE/ DRTILE problem can be found in [1, 7, 11].

**Related Work.** Both the RTILE and DRTILE problems can be solved in polynomial time when the array is one-dimensional. The RTILE problem can be solved using dynamic programming in time  $O(np)$ . For any fixed  $\epsilon < 1$ , the best known algorithm has the running time  $O(\min\{n + p^{1+\epsilon}, n \log n\})$  [8]. An extensive survey on the RTILE problem in one-dimension can be found in [8]. On the other hand, the DRTILE problem in one dimension can be solved using a greedy algorithm in linear time.

Both the RTILE and DRTILE problems have been proven to be NP-hard [7]. Grigni and Manne [6] proved that optimal  $p \times p$  tiling (which is a restricted variant of the RTILE problem) is NP-hard even when the cell weight is binary. Charikar et al. [3] showed this problem to be APX-hard and NP-hard to approximate within a factor of 2. Khanna et al. [7] proved the RTILE problem to be NP-hard to achieve a  $\frac{5}{4}$ -approximation. Recently Głuch and Loryś [5] have improved the lower bound of the RTILE problem to  $\frac{4}{3}$ . It is not known whether the binary RTILE is solvable in polynomial time or NP-hard. Khanna et al. [7] gave the first approximation algorithm for the RTILE problem with the ratio  $\frac{5}{2}$ . The approximation ratio was improved to  $\frac{7}{3}$  independently by Sharp [15] and Loryś and Paluch [9]. Loryś and Paluch [10] gave a  $\frac{9}{4}$ -approximation algorithm for this problem. Berman et al. [1] improved the approximation ratio to  $\frac{11}{5}$ . Finally, Paluch [13] gave a  $\frac{17}{8}$ -approximation for this problem and also proved that the approximation ratio is tight with respect to the used lower bound. As far as the DRTILE problem is concerned, Khanna et al. [7] gave an  $O(n^5)$ -time 4-approximation algorithm using the Hierarchical Binary Tiling (HBT) technique. They improved the approximation ratio to 2 using a modified version of the HBT technique, but the running time of this algorithm is very high making the algorithm less practical. Loryś and Paluch [9] also gave a 4-approximation for the DRTILE problem while improving the running time to linear.

The  $d$ -dimensional version of this problem was introduced by Smith and Suri [16]. They gave a  $\frac{d+3}{2}$ -approximation algorithm that runs in time  $O(n^d + p \log n^d)$ . Sharp [15] improved the approximation ratio to  $\frac{d^2+2d-1}{2d-1}$  that runs in time  $O(n^d + 2^d p \log n^d)$ . Paluch [14] gave a  $\frac{d+2}{2}$ -approximation algorithm while matching the previous running time. She also proved that the ratio is tight with respect to the known lower bound of the problem.

RPACK is an extensively studied variant of rectangle tiling, in which we are given a set of axis-parallel weighted rectangles in a  $n \times n$  grid, and the goal is to find at most  $k$  disjoint rectangles of largest weight. Khanna et al. [7] proved that this problem is NP-hard even when each rectangle intersects at most three other rectangles. They gave an  $O(\log n)$ -approximation algorithm for RPACK that runs in  $O(n^2 p \log n)$  time. In [1] Berman et al.

considered the multi-dimensional version of this problem. The dual of RPACK is known to be NP-hard even when we are interested in finding a sub-set of disjoint rectangles with a total weight equal to at least some given  $w$ . Du et al. [4] considered a min-max version of RTILE, where the weight of each tile cannot be smaller than the given lower bound and the aim is to minimize the maximum weight of a tile. They [4] gave a 5-approximation algorithm for this problem and Berman and Raskhodnikova [2] improved the approximation factor to 4 and the approximation ratio of the binary variant to 3.

**Previous Work.** The binary version of the RTILE problem has also been studied. Khanna et al. [7] gave a  $\frac{9}{4}$ -approximation for the binary RTILE problem. Lorys and Paluch [9] and Berman et al. [1] independently improved the approximation ratio for binary RTILE to 2.

**Our Results.** We improve the approximation ratio of the binary RTILE problem to  $\frac{3}{2} + \frac{p^2}{w(A)}$ , where  $w(A)$  denotes the number of ones in  $A$ . For the arrays  $A$  satisfying  $\frac{p^2}{w(A)} \approx 0$ , it implies that the approximation ratio of the algorithm amounts to  $\frac{3}{2}$ . The running time of our algorithm is linear ( $O(n^2)$ ). The approximation is best possible in the following sense. The algorithm employs the lower bound of  $L = \lceil \frac{w(A)}{p} \rceil$ , which is the only known and used bound on the optimum in all algorithms for rectangle tiling. We prove that a better approximation factor for the binary RTILE cannot be achieved using  $L$ , because there exist arrays, whose every partition contains a tile of weight at least  $(\frac{3}{2} + \frac{p}{w(A)})L$ .

The general approach to solving this problem is to some extent similar to the approach of [13]. The found tiling is also hierarchical and we use the notions of *boundaries* and *types of columns/subarrays* as well as we apply linear programming in a non-standard way. However, in the present paper the types of subarrays are organized in a somewhat different manner. In particular, the idea of *shadows* is new. To compute the desired partition of  $A$  into tiles, we only check a small number of tilings of simply defined subarrays. The subarrays are identified with the help of so called *boundaries* and their *shadows*, which, roughly speaking, designate parts of  $A$  tileable in a certain manner and having a weight greater than  $\frac{3}{2}L$ . To prove the tileability of subarrays composed of multiple simpler subarrays we employ linear programming. Its application here differs from the one in [13] in that each dimension is treated completely symmetrically and thus more “globally” and in the method of showing the feasibility of dual programs. We show that the binary DRTILE problem can be approximated by reducing it to the binary RTILE problem. As for the  $d$ -dimensional binary RTILE problem, the algorithm for the 2-dimensional binary RTILE problem can be extended to obtain an approximation for the  $d$ -dimensional binary RTILE problem. The same approximation ratio for the  $d$ -dimensional binary DRTILE problem can also be found analogously.

**Organization.** In Section 2, we recall the necessary definitions. In Section 3, we revisit the definition of a boundary and introduce shadows of a boundary. In Section 4, we assume that  $w(A) \gg p^2$  and present a  $\frac{3}{2}$ -approximation algorithm for the RTILE problem. The goal of this section is also to introduce the methods needed for the approximation of the binary RTILE more gradually, without obscuring the presentation with many technical aspects. In Section 5 we present a  $(\frac{3}{2} + \frac{p^2}{w(A)})$ -approximation algorithm for the general case (which, in particular, applies also when  $\frac{p^2}{w(A)}$  is not negligible). This approximation is achieved by applying only small modifications to the approach described in Section 4. In Section 6, we show that the approximation factor we obtain for the RTILE problem is tight with respect to the known lower bound. Section 7 contains our result on the DRTILE problem. We conclude by presenting an approximation algorithm for the multi-dimensional RTILE problem in Section 8.

## 2 Preliminaries

Let  $A$  be a two-dimensional array of size  $n \times n$ , where each of its elements belongs to the set  $\{0, 1\}$ . Given  $A$  and a natural number  $p$ , we want to partition  $A$  into  $p$  rectangular subarrays, called *tiles* so that the maximal weight of a tile is minimized. The weight of a tile  $T$ , denoted  $w(T)$ , is the sum of elements within  $T$ .  $w(A)$  denotes the weight of the whole array  $A$ . Since any array element is either equal to 0 or 1,  $w(A)$  amounts to the number of 1s in  $A$ .

First, notice that the problem is trivial when  $p \geq w(A)$ . Assume then that  $p < w(A)$ . Clearly, the maximal weight of a tile cannot be smaller than  $\frac{w(A)}{p}$ . Consequently,  $L = \lceil \frac{w(A)}{p} \rceil$  is a lower bound on the value of the optimal solution to the RTILE problem.

Thus to design an  $\alpha$ -approximation algorithm for the RTILE problem, it suffices to demonstrate the method of partitioning  $A$  into  $p$  tiles such that the weight of each tile does not surpass  $\alpha L$ .

The number  $p$  of allowed tiles is linked to the weight of the array  $A$  in the following manner.

► **Fact 1.** Let  $w(A)$  and  $L$  be as defined above. Then,  $p \geq \lceil \frac{w(A)}{L} \rceil$ .

The proof directly follows from the assumption that  $L = \lceil \frac{w(A)}{p} \rceil$

► **Definition 2.** An array  $A$  is said to be  $f$ -partitioned if it is partitioned into rectangular tiles such that the weight of any tile does not exceed  $f$ .

We denote by  $A[i]$  the  $i$ -th column of  $A$ , by  $A[i..j]$  a subarray of  $A$  consisting of columns  $i, i + 1, \dots, j$ . Thus  $A^T[i]$  denotes the  $i$ -th row of  $A$  and  $A^T[i..j]$  a subarray of  $A$  consisting of rows  $i, i + 1, \dots, j$ .

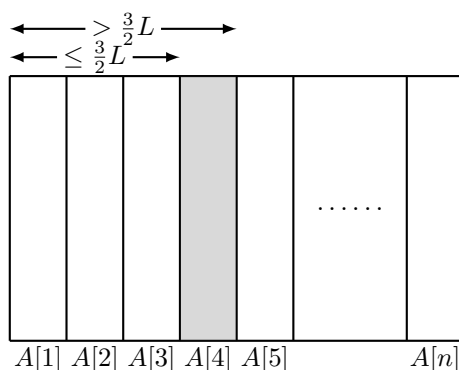
## 3 The Boundaries and Their Shadows

Let us assume that we want to design an  $\alpha$ -approximation algorithm for the RTILE problem. Hence the weight of any tile must not exceed  $\alpha L$ . In other words, we want to obtain an  $\alpha L$ -partitioning for  $A$ .

To help find such a partitioning we are going to make use of a sequence of (vertical) *boundaries* and their *shadows*. The vertical boundaries and shadows of array  $A$  are defined iteratively below. Each boundary and each shadow is a distinct column of  $A$ . The  $i$ -th boundary of  $A$  is denoted as  $B_i = A[b_i]$ , i.e.,  $B_i$  is the  $b_i$ -th column of  $A$  (or equivalently,  $B_i = A[k]$ , where  $k = b_i$ ). Similarly, the  $i$ -th shadow of  $A$  is denoted as  $B'_i = A[b'_i]$ . The number of boundaries and their shadows depends on the weight and structure of  $A$ . The shadow  $B'_i = A[b'_i]$  is equal to either  $B_i$  or the column succeeding  $B_i$ , i.e. either  $b'_i = b_i$ , or  $b'_i = b_i + 1$ . For each boundary  $B_i$  we define its *type* - we say that boundary  $B_i$  is of type  $j$ , denoted as  $t(B_i) = j$ , if its weight satisfies  $\lfloor \frac{w(B_i)}{\alpha L} \rfloor = j - 1$ .

The ideas behind boundaries and shadows are as follows. The first vertical boundary  $B_1$  indicates simply which part of the array consisting of successive columns starting from the leftmost, exceeds  $\alpha L$ . This means that such a subarray cannot be covered with one tile. However, the subarray  $A[1..b_1 - 1]$  ending on column  $b_1 - 1$  can form one tile, because its weight is not greater than  $\alpha L$ . For  $i > 1$  the  $i$ -th boundary  $B_i = A[b_i]$  is established in the following way. We distinguish two cases: (i)  $B_{i-1} = B'_{i-1}$  and (ii)  $B_{i-1} \neq B'_{i-1}$ . Let us first consider case (i). Suppose that  $t(B_{i-1}) = j$ . Any boundary of type  $j$  can be  $\alpha L$ -partitioned (horizontally) into  $j$  tiles. We check how far to the right we are able to extend one of such partitions. Thus, to identify the  $i$ th boundary  $B_i$ , we find  $b_i$  such that the subarray





■ **Figure 1** An array with column  $A[4]$  as the only boundary.

$A[b'_{i-1}..b_i - 1]$  can be  $\alpha L$ -partitioned horizontally into  $j$  tiles and the subarray  $A[b'_{i-1}..b_i]$  cannot. When (ii)  $B_{i-1} \neq B'_{i-1}$ , to identify the  $i$ th boundary  $B_i$ , we proceed in the same way as with the first boundary  $B_1$ , i.e., we find  $b_i$  such that the subarray  $A[b'_{i-1}..b_i - 1]$  can be  $\alpha L$ -partitioned horizontally into 1 tile and the subarray  $A[b'_{i-1}..b_i]$  cannot.

As for the  $i$ -th shadow  $B'_i$  we put it in the same column as the boundary  $B_i$  if the subarray  $A[b'_{i-1}..b_i]$  cannot be  $\alpha L$ -partitioned into  $t(B_i)$  tiles and otherwise, we put it just behind  $B_i$  - in column  $b_i + 1$ . Notice that in the case of a shadow we check the tileability into  $t(B_i)$  tiles and not  $t(B_{i-1})$ . Also, we observe that  $B_i \neq B'_i$  can happen only when  $t(B_i) > t(B_{i-1})$  or  $B_{i-1} \neq B'_{i-1}$ . If  $B_i \neq B'_i$ , then it means, as we later prove, that the subarray  $A[1..b_i]$  is rather easy to partition and we could in fact tile it with a proper number of tiles and start the process of tiling anew with the subarray  $A[b'_i..n]$ .

We now give a formal definition of a sequence of (vertical) *boundaries* of  $A$  and their *shadows*. For technical reasons we introduce a column  $A[0]$  to array  $A$ .

► **Definition 3.** A boundary  $B_i$  is of type  $j$ , denoted as  $t(B_i) = j$ , if its weight satisfies  $\lfloor \frac{w(B_i)}{\alpha L} \rfloor = j - 1$ . Based on that, the boundaries and their shadows are defined as follows:

1.  $B[0] = A[0], B'_0 = A[1]$ , thus  $b_0 = 0$  and  $b'_0 = 1$ ,
2.  **$i$ -th boundary  $B_i$  :**
  - a. If  $B_{i-1} = B'_{i-1}$ , then  $B_i = A[b_i]$  iff  $A[b_{i-1}..b_i - 1]$  can be  $\alpha L$ -partitioned horizontally into  $t(B_{i-1})$  tiles and  $A[b_{i-1}..b_i]$  cannot.
  - b. If  $B_{i-1} \neq B'_{i-1}$ , then  $B_i = A[b_i]$  iff  $w(A[b_{i-1} + 1..b_i - 1]) \leq \alpha L$  and  $w(A[b_{i-1} + 1..b_i]) > \alpha L$ .
3.  **$i$ -th Shadow  $B'_i$  :** Let  $t(B_i) = j$ .  
 $B'_i = B_i$  iff  $A[b'_{i-1}..b_i]$  cannot be  $\alpha L$ -partitioned horizontally into  $j$  tiles.

The horizontal boundaries are defined analogously. To illustrate the notion of boundaries and shadows let us consider a few examples.

► **Example 4.** Array  $A$  has only one vertical boundary  $B_1 = A[4]$  of type 1.

This means that the total weight of the first 3 columns does not exceed  $\alpha L$ , i.e.,  $w(A[1..3]) \leq \alpha L$ , and the weight of the subarray consisting of columns  $1 \dots 4$  does -  $w(A[1..4]) > \alpha L$ . Since  $t(B_1) = 1$ , by the definition, the weight of  $B_1 = A[4]$  is not greater than  $\alpha L$  and the shadow  $B'_1$  of  $B_1$  coincides with  $B_1$ . Since  $A$  has only one boundary, it means that the weight of the subarray consisting of all columns except for the first 3 is not greater than  $\alpha L$ , i.e.,  $w(A[4..n]) \leq \alpha L$ .

► **Example 5.** Array  $A$  has only one vertical boundary  $B_1 = A[4]$  of type 2 and  $B'_1 = A[4]$ . Exactly as in the example above, we have  $w(A[1..3]) \leq \alpha L$  and  $w(A[1..4]) > \alpha L$ . The weight of  $B_1$  satisfies:  $3L \geq w(A[4]) > \alpha L$ , because  $t(B_1) = 2$ . By the fact that  $B'_1 = B_1$ , we know that the horizontal partition of  $A[1..4]$  into 2 tiles of weight not surpassing  $\alpha L$  is impossible. Since  $A$  has only one boundary, we obtain that  $A[4..n]$  can be partitioned into  $t(B_1) = 2$  tiles.

► **Example 6.** Array  $A$  has only one vertical boundary  $B_1 = A[4]$  of type 2 and  $B'_1 = A[5]$ . Again, we have  $w(A[1..3]) \leq \alpha L$  and  $w(A[1..4]) > \alpha L$ . This time, however,  $B'_1 \neq B_1$ , therefore  $A[1..4]$  can be partitioned into 2 horizontal tiles with weight  $\alpha L$  at most. Since  $A$  has only one boundary and  $B'_1 \neq B_1$ , we have that  $w(A[5..n]) \leq \alpha L$ .

► **Lemma 7.** Let  $k$  denote the number of vertical boundaries of  $A$  and  $T_v = \sum_{i=1}^k t(B_i)$ . Then array  $A$  can be  $\alpha L$ -tilled with  $T_v + 1$  tiles.

**Proof.** Suppose first that for each  $1 \leq i \leq k$  it holds that  $B_i = B'_i$ . Then by Definition 3, each subarray  $A[b_i..b_{i+1} - 1]$  can be tiled horizontally with  $t(B_i)$  tiles and the subarray  $A[1..b_1 - 1]$  can be covered by 1 tile. Therefore we indeed use  $T_v + 1$  tiles.

For the general case, let  $i = \min\{k : B_k \neq B'_k\}$ . It means that the subarray  $A[b_{i-1}..b_i]$  can be tiled horizontally with  $t(B_i)$  tiles. By Definition 3 for each  $j \leq i - 2$  the subarray  $A[b_j..b_{j+1} - 1]$  can be tiled horizontally with  $t(B_j)$  tiles and the subarray  $A[1..b_1 - 1]$  can be covered by a single tile. This way the number of used tiles amounts to  $\sum_{j=1}^{i-2} t(B_j) + t(B_i) + 1 \leq \sum_{j=1}^i t(B_j)$ . We continue in the same manner with the subarray  $A[b'_i..n]$ . ◀

Analogously, we define a horizontal sequence of boundaries of  $A$ , i.e., a vertical sequence of boundaries of  $A^T$ .

Throughout the paper,  $B_1, B_2, \dots, B_k$  and  $C_1, \dots, C_l$  denote, respectively, the vertical and horizontal sequence of boundaries of  $A$ . Let  $T_v = \sum_{i=1}^k t(B_i)$  and  $T_h = \sum_{i=1}^l t(C_i)$  and let  $T = \min\{T_v, T_h\}$ .

► **Fact 8.** Array  $A$  can be  $\alpha L$ -tilled with  $T + 1$  tiles.

Since we can always  $\alpha L$ -partition  $A$  into  $T + 1$  tiles, to prove that there exists an  $\alpha$ -approximation algorithm for the binary RTILE problem, it suffices to show that  $T + 1$  is an allowed number of tiles, i.e., that  $T + 1 \leq p$ . To do so, it is enough to prove that it always holds that  $w(A) > TL$ . This is because since  $w(A) \leq pL$  and  $T$  and  $p$  are integers, the inequality  $w(A) > TL$  implies  $T + 1 \leq p$ .

We state this observation as:

► **Fact 9.** Let  $\alpha$  be such that for any  $A$  it holds that  $w(A) > TL$ . Then  $p \geq T + 1$  and there exists an  $\alpha$ -approximation algorithm for the binary RTILE problem.

Let us first note that it is easy to prove that  $w(A) > \frac{TL}{2}$ .

► **Lemma 10.** The weight of  $A$  satisfies  $w(A) > \frac{TL}{2}$ . Hence,  $p > \frac{T}{2}$ .

**Proof.** We begin by proving that it is always possible to partition the array  $A$  vertically into disjoint subarrays  $A_1, A_2, \dots, A_k$  where each subarray except the first ( $A_1$ ) will contain one of the following boundary types:

1. A single boundary of type  $j$ , where  $j$  is greater than one.
2. Two boundaries of type 1.
3. A boundary of type  $j$  followed by a boundary of type 1, where  $j$  is greater than one.

The first subarray  $A_1$  may have any of the types of boundaries mentioned above or a single boundary of type 1. We will now describe how to construct these subarrays.

Let  $B_1, B_2, \dots, B_l$  be a sequence of vertical boundaries of  $A$ . We construct the subarrays  $A_1, \dots, A_k$  iteratively. If  $t(B_l) > 1$ , then we define  $A[b_l..n]$  as the last vertical subarray, otherwise, the last subarray is represented by  $A[b_{l-1}..n]$ . We then repeat this process on the remaining array  $A[1..b_{x-1}]$ , where  $x \in \{l-1, l\}$  based on our choice of the last vertical subarray. We continue until we cannot construct a vertical subarray with one of the sets of boundaries mentioned in points 1 – 3.

In this case, the remaining subarray either has no boundary or has a boundary of type 1. If it has no boundary, we merge it with the vertical subarray containing the first boundary. If it has a boundary of type 1, we define it as the first vertical subarray and call it  $A_1$ .

Suppose  $T_i$  represents the sum of the types of boundaries that are located within the subarray  $A_i$ . Our goal is to prove that  $w(A_i) \geq \frac{1}{2}T_iL$ . If  $A_i$  contains a boundary  $B_r$  of type  $j > 1$  then,

$$\begin{aligned} w(A_i) &\geq w(B_r) \geq (j-1) \cdot \frac{3}{2}L \geq \frac{1}{3}(j+1) \cdot \frac{3}{2}L \\ &\geq \frac{1}{2}t(B_r)L = \frac{1}{2}T_iL \end{aligned}$$

If  $A_i$  contains two boundaries  $B_r$  and  $B_{r+1}$  of type 1 then,

$$\begin{aligned} w(A_i) &\geq w(A[b_r \dots b_{r+1}]) \geq \frac{3}{2}L \\ &\geq \frac{1}{2}(t(B_r) + t(B_{r+1})) \frac{3}{2}L > \frac{1}{2}T_i \cdot L \end{aligned}$$

If  $B_r$  is a boundary of type  $j > 1$  and  $B_{r+1}$  is a boundary of type 1 in  $A_i$  then,

$$\begin{aligned} w(A_1) &\geq w(B_r) \geq (j-1) \cdot \frac{3}{2}L \geq \frac{1}{3}(j+1) \cdot \frac{3}{2}L \\ &\geq \frac{1}{2}(t(B_r) + t(B_{r+1}))L = \frac{1}{2}T_i \cdot L \end{aligned}$$

If the  $A_1$  contains one of the above sets of boundaries then  $w(A_1) \geq \frac{T_1 \cdot L}{2}$ . Otherwise,  $A_1$  contains a single boundary of type 1. Then,

$$w(A_1) \geq \frac{3}{2}L > \frac{1}{2}t(B_1)L = \frac{1}{2}T_1 \cdot L$$

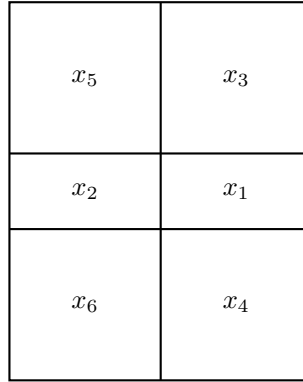
In conclusion we have proved that for each  $A_i$ ,  $w(A_i) \geq \frac{T_i \cdot L}{2}$ . Therefore  $w(A) \geq \frac{TL}{2}$ . ◀

We now present a lemma that establishes conditions under which a subarray  $A'$  of  $A$  can be partitioned into horizontal tiles.

► **Lemma 11.** *Let  $A' = A[i_1 \dots i_2]$  be a subarray of  $A$  and  $k$  a natural number greater or equal 2.*

*Suppose that  $\frac{w(A[i_1]) + w(A[i_2])}{k} + w(A[i_1 + 1 \dots i_2 - 1]) \leq \alpha L$ . Then  $A'$  can be partitioned into  $k$  horizontal tiles of weight at most: (i)  $\alpha L + 1$  if  $k = 2$ , (ii)  $\alpha L + 2$  if  $k > 2$ .*

**Proof.** We divide the number equal to the sum of the weights of two columns  $A[i_1]$  and  $A[i_2]$  (the columns may be unconnected) by  $k$ . We check where the division lines fall with respect to the subarray. Often they may occur in the middle of a row (consisting of two array elements) and we have to move the division so that the whole row is included or the



■ **Figure 2** An array with one horizontal boundary containing the subarrays  $x_2$  and  $x_1$  and one vertical boundary containing the subarrays  $x_3$ ,  $x_1$  and  $x_4$ .

whole row is excluded. If  $k = 2$ , then we choose one of the two options - moving the division upwards or downwards, hence, in the worst case we may have to increase the weight of one tile by 1. For  $k > 2$ , we may have to shift the division by almost the whole row and thus increase the weight of some tiles by 2. Next we extend this partition to include the subarray  $A'' = A[i_1 + 1 \dots i_2 - 1]$  - we do not change the partition of the two-column subarray, but simply follow the partition lines. In the worst case the whole weight of  $A''$  will fall into only one tile - yielding a tile of weight  $\frac{w(A[i_1]) + w(A[i_2])}{k} + w(A[i_1 + 1 \dots i_2 - 1])$ . ◀

**4** A  $\frac{3}{2}$ -approximation when  $w(A) \gg p^2$

In this section we deal with arrays such that  $\frac{p^2}{w(A)}$  is close to 0, which means that the total weight of any  $p^2$  elements of  $A$  is negligible. We are going to show that under this assumption, for  $\alpha = \frac{3}{2}$ , the weight of  $A$  satisfies  $w(A) > TL$ . Hence, by Fact 9 we get that for this type of arrays there exists a  $\frac{3}{2}$ -approximation for the binary RTILE problem. For the general case the proof that  $\alpha = \frac{3}{2} + \frac{p^2}{w(A)}$ , the weight of  $A$  satisfies  $w(A) > TL$ , presented in the next section will be only a slight modification of the one shown here.

► **Convention 12.** *Throughout this section whenever we speak about tiling and partitioning, we respectively mean “ $\frac{3}{2}L$ -partitioning” and “tiling using tiles of weight at most  $\frac{3}{2}L$ ”.*

► **Remark.** The total number of cells at the intersection of the horizontal and vertical boundaries is  $O(T^2)$ . By Lemma 10 we have that  $O(T^2) = O(p^2)$ . Therefore by the assumption of this section, it implies that the total weight of the cells in the intersections of the boundaries is negligible with respect to the total weight of the array.

► **Observation 13** ([12]). *Assume we have two complexes: one as in Figure 2 and the other with the variables related to the variables of the first one as follows:  $x'_1 = x_1$ ,  $x'_4 = x_4$ ,  $x'_3 = x_3$ ,  $x'_5 = x'_6 = 0$  and  $x'_2 = x_2 + \max\{x_5, x_6\}$ . Then the weight of the second complex is not bigger than the weight of the first one while the inequalities describing the first complex remain true for the second.*

Given an array  $A$  we build a linear program, with the help of which we will be able to relate the total weight of the array to the sum of types of boundaries  $T$ , i.e., we will show that  $w(A) > TL$ .

Using Observation 13, we can assume that the whole weight of the array  $A$  is contained in the boundaries, i.e., each element of  $A$  that does not belong to any boundary has value 0. Each vertical boundary  $B_i$  is crossed by  $l$  horizontal boundaries and thus cut into  $l + 1$  parts. We assign a variable  $x_{j,i}$  to each part, i.e., the  $j$ th part of  $B_i$  consists of elements  $A[c_{j-1} + 1, b_i], A[c_{j-1} + 2, b_i], \dots, A[c_j - 1, b_i]$  and  $x_{j,i}$  denotes the sum of the weights of these elements. Similarly, each horizontal boundary  $C_i$  is crossed by  $k$  vertical boundaries and thus cut into  $k + 1$  parts. We assign a variable  $z_{i,j}$  to each such part. The value of each variable  $x_{j,i}$  or  $z_{i,j}$  denotes the weight of the corresponding part of the boundary.

In the linear program, we minimize the sum of non-negative variables  $x_{j,i}$  and  $z_{i,j}$  subject to a set of constraints associated with the boundaries. For each vertical boundary  $B_i$  we will have either one or two constraints of the following form:

1. If  $t(B_i) > 1$ , then we add the constraint  $\frac{1}{t(B_i)-1} \sum_{j=1}^{l+1} x_{j,i} \geq \frac{3}{2}L$ , which simply describes the total weight of  $B_i$ .
2. a.  $B'_{i-1} \neq B_{i-1}$ .
  - i.  $t(B_i) = 1$ . The added constraint is  $\sum_{j=1}^l z_{j,i} + \sum_{j=1}^{l+1} x_{j,i} > \frac{3}{2}L$ .
  - ii.  $t(B_i) > 1$  and  $B'_i = B_i$  (which means that  $A[b'_{i-1}..b_i]$  cannot be tiled horizontally with  $t(B_i)$  tiles). By Lemma 11 we are justified to add the constraint  $\sum_{j=1}^l z_{j,i} + \frac{1}{t(B_i)} \sum_{j=1}^{l+1} x_{j,i} > \frac{3}{2}L$ .
  - iii.  $t(B_i) > 1$  and  $B'_i \neq B_i$ . In this case we do not add any constraint.
- b.  $B'_{i-1} = B_{i-1}$ .
  - i.  $B'_i \neq B_i$ . The added constraint is  $\sum_{j=1}^l z_{j,i} + \frac{1}{t(B_{i-1})} (\sum_{j=1}^{l+1} x_{j,i} + \sum_{j=1}^{l+1} x_{j,i-1}) > \frac{3}{2}L$ .
  - ii.  $B'_i = B_i$ . Let  $T_i = \max\{t(B_{i-1}), t(B_i)\}$ . The constraint we add is  $\sum_{j=1}^l z_{j,i} + \frac{1}{T_i} (\sum_{j=1}^{l+1} x_{j,i} + \sum_{j=1}^{l+1} x_{j,i-1}) > \frac{3}{2}L$ . The constraint is a consequence of Lemma 11.

Thus, each  $B_i$  defines either one or two constraints. Analogously, each horizontal variable  $C_j$  also defines one or two constraints. The linear program dual to the one we have just described has dual variables  $y'_i, y_i$ . For each  $B_i$  with  $t(B_i) > 1$  let  $y'_i$  denote the dual variable corresponding to the constraint  $\frac{1}{t(B_i)-1} \sum_{j=1}^{l+1} x_{j,i} > \frac{3}{2}L$ . The other type of a constraint (if it exists) defined by  $B_i$  is represented by  $y_i$ . The dual variables corresponding to horizontal boundaries are  $w_i, w'_i$ .

► **Example 14.** In this example array  $A$  has one vertical boundary  $B_1$  of type 1 and one horizontal boundary  $C_1$  of type 1.

|           |           |           |       |
|-----------|-----------|-----------|-------|
| 0         | $x_{1,1}$ | 0         | $C_1$ |
| $z_{1,1}$ |           | $z_{1,2}$ |       |
| 0         | $x_{2,1}$ | 0         |       |
| $B_1$     |           |           |       |

■ **Figure 3** An array with one vertical and one horizontal boundary.

The linear program for  $A$  looks as follows. In brackets we give the dual variables corresponding to respective inequalities.

$$\begin{aligned}
 &\text{minimize} && x_{1,1} + x_{2,1} + z_{1,1} + z_{1,2} \\
 &\text{subject to} && x_{1,1} + x_{2,1} + z_{1,1} > \frac{3}{2}L \quad (y_1) \\
 &&& x_{1,1} + z_{1,1} + z_{1,2} > \frac{3}{2}L \quad (w_1)
 \end{aligned}$$

28:10 Rectangle Tiling Binary Arrays

|  |           |           |           |           |           |       |
|--|-----------|-----------|-----------|-----------|-----------|-------|
|  | 0         | $x_{1,1}$ | 0         | $x_{1,2}$ | 0         |       |
|  | $z_{1,1}$ |           | $z_{1,2}$ |           | $z_{1,3}$ | $C_1$ |
|  | 0         | $x_{2,1}$ | 0         | $x_{2,2}$ | 0         |       |
|  | $z_{2,1}$ |           | $z_{2,2}$ |           | $z_{2,3}$ | $C_2$ |
|  | 0         | $x_{3,1}$ | 0         | $x_{3,2}$ | 0         |       |
|  |           | $B_1$     |           | $B_2$     |           |       |

■ **Figure 4** An array with two vertical and two horizontal boundaries.

► **Example 15.** In this example array  $A$  has two vertical boundaries and two horizontal ones, each of the four boundaries is of type 1. The array is depicted in Figure 4. The linear program for  $A$  looks as follows. In brackets we give the dual variables corresponding to respective inequalities.

$$\begin{aligned}
 &\text{minimize} && \sum_{i=1}^2 \sum_{j=1}^3 x_{j,i} + \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} \\
 &\text{subject to} && \sum_{j=1}^3 x_{j,1} + \sum_{i=1}^2 z_{i,1} > \frac{3}{2}L \quad (y_1) \\
 &&& \sum_{i=1}^2 \sum_{j=1}^3 x_{j,i} + \sum_{i=1}^2 z_{i,2} > \frac{3}{2}L \quad (y_2) \\
 &&& \sum_{j=1}^3 z_{1,j} + \sum_{i=1}^2 x_{1,i} > \frac{3}{2}L \quad (w_1) \\
 &&& \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} + \sum_{i=1}^2 x_{2,i} > \frac{3}{2}L \quad (w_2)
 \end{aligned}$$

► **Example 16.** In this example array  $A$  has two vertical boundaries  $B_1, B_2$  and two horizontal ones  $C_1, C_2$ . Their types are the following:  $t(B_1) = t(C_2) = 2$  and  $t(B_2) = t(C_1) = 1$ . Also  $B'_1 \neq B_1$  and  $C'_2 = C_2$ .

$$\begin{aligned}
 &\text{minimize} && \sum_{i=1}^2 \sum_{j=1}^3 x_{j,i} + \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} \\
 &\text{subject to} && \frac{1}{2} \sum_{j=1}^3 x_{j,1} + \sum_{i=1}^2 z_{i,1} > \frac{3}{2}L \quad (y_1) \\
 &&& \sum_{j=1}^3 x_{j,1} > \frac{3}{2}L \quad (y'_1) \\
 &&& \sum_{i=1}^2 z_{i,2} + \sum_{j=1}^3 x_{j,2} > \frac{3}{2}L \quad (y_2) \\
 &&& \sum_{j=1}^3 z_{1,j} + \sum_{i=1}^2 x_{1,i} > \frac{3}{2}L \quad (w_1) \\
 &&& \frac{1}{2} \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} + \sum_{i=1}^2 x_{2,i} > \frac{3}{2}L \quad (w_2) \\
 &&& \sum_{j=1}^3 z_{2,j} > \frac{3}{2}L \quad (w'_2)
 \end{aligned}$$

Let us now build dual linear program for the primal linear program of the Example 16. The dual linear program has the form:

$$\begin{aligned}
 &\text{maximize} && \frac{3}{2}L(y_1 + y_2 + w_1 + w_2) \\
 &\text{subject to} && y'_1 + \frac{1}{2}y_1 + w_1 \leq 1 \quad (x_{1,1}) \\
 &&& y_2 + w_1 \leq 1 \quad (x_{1,2}) \\
 &&& y'_1 + \frac{1}{2}y_1 + w_2 \leq 1 \quad (x_{2,1}) \\
 &&& y_2 + w_2 \leq 1 \quad (x_{2,2}) \\
 &&& y_1 + w_1 + \frac{1}{2}w_2 \leq 1 \quad (z_{1,1}) \\
 &&& y_2 + w_1 + \frac{1}{2}w_2 \leq 1 \quad (z_{1,2}) \\
 &&& y_1 + w'_2 + \frac{1}{2}w_2 \leq 1 \quad (z_{2,1}) \\
 &&& y_2 + w'_2 + \frac{1}{2}w_2 \leq 1 \quad (z_{2,2})
 \end{aligned}$$

■ **Algorithm 1** for the binary RTILE problem.

---

```

1:  $A \leftarrow [1 \dots n, 1 \dots n]$  a two-dimensional array
2: Construct the horizontal and vertical boundaries and their shadows using Definition 3.
3:  $B \leftarrow \{B_1, B_2, \dots, B_k\}$  (the vertical boundaries, where each  $B_i = A[b_i]$ )
4:  $B' \leftarrow \{B'_1, B'_2, \dots, B'_k\}$  (the shadows of  $B_i$ s, where each  $B'_i = A[b'_i]$ )
5:  $t(B) \leftarrow \{t(B_1), t(B_2), \dots, t(B_k)\}$  (the types of the vertical boundaries)
6:  $C \leftarrow \{C_1, C_2, \dots, C_l\}$  (the horizontal boundaries, where each  $C_i = A[c_i]$ )
7:  $C' \leftarrow \{C'_1, C'_2, \dots, C'_l\}$  (the horizontal boundaries, where each  $C'_i = A[c'_i]$ )
8:  $t(C) \leftarrow \{t(C_1), t(C_2), \dots, t(C_l)\}$  (the types of the horizontal boundaries)
9:  $T_v \leftarrow \sum_{i=1}^k t(B_i)$ 
10:  $T_h \leftarrow \sum_{i=1}^l t(C_i)$ 
11: if  $T_v \leq T_h$  then
12:   use the vertical boundaries  $B$  as described below from line 15
13: else
14:   use the horizontal boundaries  $C$  instead of the vertical ones
15: if  $B_k = B'_k$  then
16:   partition  $A[b_k \dots n]$  horizontally into  $t(B_k)$  tiles
17: else
18:   cover  $A[b_k + 1 \dots n]$  with one tile
19: for  $i = k - 1 \dots 1$  do
20:   if  $B_i = B'_i$  then
21:     if  $B_{i+1} = B'_{i+1}$  then
22:       tile  $A[b_i \dots b_{i+1} - 1]$  horiz. into  $t(B_i)$  tiles (by point 2a of Definition 3)
23:     else
24:       partition  $A[b_i \dots b_{i+1}]$  horiz. into  $t(B_{i+1})$  tiles (by point 3 of Definition 3)
25:   else  $(B_{i+1} \neq B'_{i+1})$ 
26:     if  $B_{i+1} = B'_{i+1}$  then
27:       cover  $A[b_i + 1 \dots b_{i+1} - 1]$  horiz. with one tile (by point 2b of Definition 3)
28:     else
29:       partition  $A[b_i + 1 \dots b_{i+1}]$  horiz. into  $t(B_{i+1})$  tiles (by point 3 of Definition 3)
30: if  $B_1 = B'_1$  then
31:   cover  $A[1 \dots b_1 - 1]$  with one tile
32: else
33:   partition  $A[1 \dots b_1]$  horiz. into  $t(B_1)$  tiles

```

---

To figure out the form of constraints constituting the dual program in general, let us consider a variable  $x_{j,i}$ . Notice that it occurs in at most one constraint defined by a horizontal boundary. It can possibly be contained only in the constraint defined by  $C_j$  represented by  $w_j$ , where its coefficient is 1. If  $t(B_i) = 1$ , then we do not have  $y'_i$  and  $x_{j,i}$  occurs in the constraint represented by  $y_i$  and possibly in the constraint represented by  $y_{i+1}$ . Thus the inequality in the dual program corresponding to  $x_{j,i}$  has the form  $\alpha_{j,i}y_i + \alpha_{j,i+1}y_{i+1} + \beta_{j,i}w_j \leq 1$ , where each of the coefficients belongs to  $[0, 1]$ .

If  $t(B_i) > 1$ , then we do have  $y'_i$  and  $x_{j,i}$  occurs in this constraint with the coefficient  $\frac{1}{t(B_i)-1}$ . If  $B_i \neq B'_i$ , then  $x_{j,i}$  does not occur in any other constraints and the inequality in the dual program has the form  $\frac{1}{t(B_i)-1}y'_i + \beta_{j,i}w_j \leq 1$ , where  $\beta_{j,i} \in [0, 1]$ . Otherwise,  $x_{j,i}$  may also belong to the constraints represented by  $y_i$  and  $y_{i+1}$ . In each one of them it occurs with the coefficient equal to at most  $\frac{1}{t(B_i)}$ . Therefore the inequality has the form  $\frac{1}{t(B_i)-1}y'_i + \alpha_{j,i}y_i + \alpha_{j,i+1}y_{i+1} + \beta_{j,i}w_j \leq 1$ , where each of the coefficients  $\alpha_{j,i+1}, \alpha_{j,i}$  belongs to  $[0, \frac{1}{t(B_i)}]$ .

We are ready to lower bound the weight of the array  $A$  with the aid of its boundaries and their shadows.

► **Lemma 17.**  $w(A) > TL$ .

## 28:12 Rectangle Tiling Binary Arrays

**Proof.** Since the value of any cost function of the dual linear program described above is a lower bound on the minimal value of the cost function of the primal linear program, it suffices to find a feasible assignment of the dual variables such that the cost function will be have value greater than  $TL$ .

▷ **Claim 18.** We can satisfy all constraints of the dual program by assigning the following values to the dual variables. Each  $y_i$  and each  $w_j$  is assigned  $\frac{1}{3}$ . If  $B_i$  defines only one constraint  $y'_i$ , then we assign  $\frac{t(B_i)}{3}$  to  $y'_i$ . Otherwise  $y'_i$  is assigned  $\frac{t(B_i)-1}{3}$ .

The claim follows from the fact that the inequality  $\frac{1}{T-1} \cdot \frac{T}{3} + \frac{1}{3} \leq 1$  is satisfied by each  $T \geq 2$  and that the inequality  $\frac{T-1}{3(T-1)} + \frac{2}{3T} + \frac{1}{3} \leq 1$  is also satisfied by each  $T \geq 2$ .

This means that the total value contributed by the dual variables  $y_i, y'_i$  (corresponding to constraints defined by boundary  $B_i$ ) is at least  $\frac{t(B_i)}{3}$ .

Thus  $w(A) > \frac{3}{2}L(\frac{T_v}{3} + \frac{T_h}{3}) \geq TL$ . ◀

We show a method for finding a tiling of  $A$  with at most  $T + 1$  tiles. By Facts 1, 8, Lemmas 7 and 17, we proved that  $T + 1 \leq p$ , when the approximation factor is  $\frac{3}{2}$ . In other words, we obtain a  $\frac{3}{2}$ -approximation algorithm for binary RTILE.

► **Theorem 19.** For any array  $A$  satisfying  $\frac{p^2}{w(A)} \approx 0$ , there exists a linear time  $\frac{3}{2}$ -approximation algorithm for binary RTILE.

### 5 A $(\frac{3}{2} + \beta)$ -approximation

In this section we examine the general case, arrays such that  $\frac{p^2}{w(A)}$  is not negligible. We will aim for a  $(\frac{3}{2} + \beta)$ -approximation. When  $\beta < \frac{1}{2}$ , the approximation ratio of our algorithm is better than 2. Throughout the section, whenever we refer to tiling and partitioning, we mean  $(\frac{3}{2} + \beta)L$ -partitioning and tiling using tiles of weight at most  $(\frac{3}{2} + \beta)L$ .

We define a sequence of boundaries and shadows analogously as in the previous section, but with respect to  $(\frac{3}{2} + \beta)L$ , i.e., we replace each occurrence of “ $\frac{3}{2}L$ ” with “ $(\frac{3}{2} + \beta)L$ ” and modify the meaning of tiling and partitioning accordingly, i.e., to  $(\frac{3}{2} + \beta)L$ -partitioning.

We want to prove an analogue of Lemma 17. To this end we will consider an analogous linear program, in which we have all the variables occurring in the previous section and additionally we have a variable  $s_{i,j}$  for each pair  $(B_i, C_j)$ , which denotes the element of  $A$  at the intersection of the vertical boundary  $B_i$  and the horizontal boundary  $C_j$ . The function we minimize is  $\sum_{i=1}^k x_{j,i} + \sum_{j=1}^l z_{i,j} + \sum_{i=1}^k \sum_{j=1}^l s_{i,j}$ . For each variable  $s_{i,j}$  we have an additional constraint:  $-s_{i,j} \geq -1$ . The variable  $s_{i,j}$  is also included in all those constraints which refer to the part of  $A$  covering the intersection of  $B_i$  with  $C_j$ .

For instance, the linear program for the array from Example 14 is modified as follows:

$$\begin{aligned} & \text{minimize} && x_{1,1} + x_{2,1} + z_{1,1} + z_{1,2} + s_{1,1} \\ & \text{subject to} && x_{1,1} + x_{2,1} + z_{1,1} + s_{1,1} > (\tfrac{3}{2} + \beta)L & (y_1) \\ & && x_{1,1} + z_{1,1} + z_{1,2} + s_{1,1} > (\tfrac{3}{2} + \beta)L & (w_1) \\ & && -s_{1,1} \geq -1 & (t_{1,1}). \end{aligned}$$

The linear program for the array from Example 15 in the new scenario looks as follows:

$$\begin{aligned} & \text{min} && \sum_{i=1}^2 \sum_{j=1}^3 x_{j,i} + \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} + \sum_{i=1}^2 \sum_{j=1}^2 s_{i,j} \\ & \text{s.t.} && \sum_{j=1}^3 x_{j,1} + \sum_{i=1}^2 z_{i,1} + \sum_{j=1}^2 s_{1,j} > (\tfrac{3}{2} + \beta)L & (y_1) \\ & && \sum_{i=1}^2 \sum_{j=1}^3 x_{j,i} + \sum_{i=1}^2 z_{i,2} + \sum_{i=1}^2 \sum_{j=1}^2 s_{i,j} > (\tfrac{3}{2} + \beta)L & (y_2) \\ & && \sum_{j=1}^3 z_{1,j} + \sum_{i=1}^2 x_{1,i} + \sum_{i=1}^2 s_{i,1} > (\tfrac{3}{2} + \beta)L & (w_1) \\ & && \sum_{i=1}^2 \sum_{j=1}^3 z_{i,j} + \sum_{i=1}^2 x_{2,i} + \sum_{i=1}^2 \sum_{j=1}^2 s_{i,j} > (\tfrac{3}{2} + \beta)L & (w_2) \\ & && -s_{i,j} \geq -1(t_{i,j}), \text{ for each } 1 \leq i, j \leq 2. \end{aligned}$$



Correspondingly, in the dual program we maximize  $(\frac{3}{2} + \beta)L(\sum y_i + \sum w_j) - \sum t_{i,j}$  and we have an additional constraint for each primal variable  $s_{i,j}$ .

The dual linear program for the array from Example 15 contains the following additional inequalities.

$$\begin{aligned} y_1 + y_2 + w_1 + w_2 - t_{1,1} &\leq 1 && (s_{1,1}) \\ y_1 + y_2 + w_2 - t_{1,2} &\leq 1 && (s_{1,2}) \\ y_2 + w_1 + w_2 - t_{2,1} &\leq 1 && (s_{2,1}) \\ y_2 + w_2 - t_{2,2} &\leq 1 && (s_{2,2}). \end{aligned}$$

We can see that if we want to assign  $\frac{1}{3}$  to each variable  $y_i, w_j$ , then we sometimes also have to assign  $\frac{1}{3}$  to variables  $t_{i,j}$  to ensure the feasibility - compare the first inequality in the set of additional inequalities above. We can notice that we have to assign  $\frac{1}{3}$  to  $t_{i,j}$  only if both  $i < k$  and  $j < l$ , i.e. when  $s_{i,j}$  does not belong to  $B_k$  or  $C_l$ .

► **Lemma 20.** For  $\beta = \frac{p^2}{w(A)}$ , it holds that  $w(A) > TL$ .

**Proof.** We can satisfy all constraints of the dual program by assigning the following values to the dual variables. Each  $y_i$  and each  $w_j$  is assigned  $\frac{1}{3}$ . If  $B_i$  defines only one constraint  $y'_i$ , then we assign  $\frac{t(B_i)}{3}$  to  $y'_i$ . Otherwise  $y'_i$  is assigned  $\frac{t(B_i)-1}{3}$ . Also, each  $t_{i,j}$  such that  $i < k$  and  $j < l$  is assigned  $\frac{1}{3}$ .

Some of the constraints in the primal program have value  $(\frac{3}{2} + \beta)L - 2$  on the right hand side. Such constraints correspond to some borders of type greater than 2, when we use Lemma 11. Let us analyze such cases in more detail. Assume that for a boundary  $B_i$  of type  $k > 2$  we indeed use Lemma 11. Then the primal linear program contains a constraint with value  $(\frac{3}{2} + \beta)L - 2$  on the right hand side. This constraint corresponds to the dual variable  $y_i$ . We notice that  $B_i$  also defines a constraint of type 1, which has  $(\frac{3}{2} + \beta)L$  on the right hand side and corresponds to the dual variable  $y'_i$ . Hence each such boundary contributes at least  $\frac{t(B_i)-1}{3}(\frac{3}{2} + \beta)L + \frac{1}{3}(\frac{3}{2} + \beta)L - 2 \geq (\frac{(\frac{3}{2} + \beta)L - \frac{2}{3}t(B_i)}{3})$  to the cost function of the dual linear program.

Similarly, some of the constraints in the primal program have value  $(\frac{3}{2} + \beta)L - 1$  on the right hand side. Such constraints correspond to some borders of type equal to 2, when we use Lemma 11. Each such boundary contributes at least  $(\frac{(\frac{3}{2} + \beta)L - \frac{1}{2}t(B_i)}{3})$  to the cost function of the dual linear program.

Thus the value of the cost function of the dual linear program is lower bounded by  $((\frac{3}{2} + \beta)L - \frac{2}{3})(\frac{T_v}{3} + \frac{T_h}{3}) - \frac{(T_h-1)(T_v-1)}{3} \geq TL + p\frac{T_v+T_h}{3} - \frac{2}{3}(T_v + T_h) - \frac{(T_h-1)(T_v-1)}{3}$ . Since  $L \geq \frac{w(A)}{p}$ , we get that  $w(A) \geq TL + p\frac{T_v+T_h}{3} + \frac{T_v+T_h}{9} - \frac{T_vT_h}{3} - \frac{1}{3}$ . Because  $p \geq T$ , we obtain that  $\frac{p(T_v+T_h)}{3} + \frac{T_v+T_h}{9} - \frac{T_vT_h}{3} - \frac{1}{3} \geq \frac{T^2}{3} + \frac{T_v+T_h}{9} - \frac{1}{3} > 0$ .

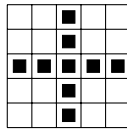
Therefore,  $w(A) > TL$ . ◀

► **Theorem 21.** There exists a  $(\frac{3}{2} + \frac{p^2}{w(A)})$ -approximation algorithm for binary RTILE that has a linear  $(O(n^2))$  running time.

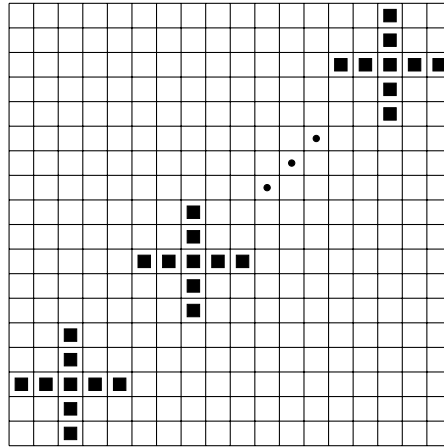
## 6 Tightness of approximation

In this section, we show that the approximation ratio for the RTILE problem is tight with respect to the only known lower bound. Precisely, we prove the following theorem.

► **Theorem 22.** Let  $p = 2k$ , for some  $k \in \mathbb{N}$ . Then, there exists a binary array  $A_k$  such that the maximum weight of a tile in any tiling of  $A_k$  into  $p$  tiles has weight at least  $\frac{3}{2} \cdot \frac{w(A_k)}{p} + 1$ .



(a)



(b)

■ **Figure 5** The empty squares denote a value of 0, while the ones are colored black. (a) On tiling this array with 3 tiles, one tile will always contain 5 ones, giving an approximation factor of  $\frac{5}{3}$ . (b) 4-crosses placed in an array for proving an approximation lower bound of  $\frac{3}{2}$ .

We define an  $L$ -cross to consist of  $2L + 1$  ones, it is obtained by taking a  $(L + 1) \times (L + 1)$  array, and filling the  $(\frac{L}{2} + 1)^{th}$  row as well as the  $(\frac{L}{2} + 1)^{th}$  column with ones, finally the rest of the entries are filled with zeros. The coordinate  $(\frac{L}{2} + 1, \frac{L}{2} + 1)$  is referred to as the center of the  $L$ -cross defined above. An  $L$ -cross centered at  $(x, y)$  is obtained by translating the center of an  $L$ -cross to the coordinate  $(x, y)$ . Note that an  $L$ -cross consists of four contiguous segments of ones, referred to as *arms*, each containing  $\frac{L}{2}$  ones. We define  $A_k$  as shown in Figure 3.

**Proof.** Suppose that  $p$  is even, therefore  $p = 2k$ , for some  $k \in \mathbb{N}$ ; our input binary array  $A_k$  is obtained as follows. We place  $k$  many  $L$ -crosses centered at  $(j \cdot (L + 1) - \frac{L}{2}, j \cdot (L + 1) - \frac{L}{2})$ , for each  $1 \leq j \leq k$ , the rest of the entries of  $A_k$  are zero. Note that the  $L$ -crosses are placed diagonally in a non-overlapping manner, and  $\frac{w(A)}{p} = L$ . The array for  $L = 4$  is illustrated in Figure 5(a).

If  $p = 2$ , then it is obvious that one tile will have to contain 3 arms of the cross and thus have weight  $\frac{3}{2}L + 1$ . We will prove that for every  $k \in \mathbb{N}$ , one of the tiles will have weight at least  $\frac{3}{2}L + 1$ .

Suppose that for  $k$  crosses and  $2k$  tiles the thesis holds by induction. We will now prove it for  $k + 1$  crosses and  $2k + 2$  tiles. Let  $T_1$  be the tile that contains the cell  $A_{k+1}[1, 1]$ . If this tile has weight smaller than  $\frac{3}{2}L$ , then we have the following two cases:

1. If  $T_1$  does not contain the center of the lower left cross  $A_{k+1}[\frac{L}{2} + 1, \frac{L}{2} + 1]$ , then  $T_1$  is formed either by the first  $\frac{L}{2}$  columns or the first  $\frac{L}{2}$  rows. Due to symmetry, it is enough to consider the case when  $T_1$  is formed by the first  $\frac{L}{2}$  columns. Let  $T_2$  be the tile that contains  $A_{k+1}[\frac{L}{2} + 1, 1]$ . If  $T_2$  has weight smaller than  $\frac{3}{2}L$ , then its upper right corner

$A_{k+1}[x, y]$  is such that either  $x < L + 1$  or  $y < L + 1$ . Due to symmetry, it is enough to consider the case when  $x < L + 1$ . In this case, if  $y < n$ , then we can extend  $T_2$  so that  $y = n$  without increasing the weight of  $T_2$  as it would not intersect any new  $L$ -cross.

Thus, we are left with  $2k$  tiles, and an array which has  $A_k$  as a subarray, therefore by the induction hypothesis we get that the weight of maximum weight tile is at least  $\frac{3}{2}L + 1$ .

2. If  $T_1$  contains the center of the cross  $A_{k+1}[\frac{L}{2} + 1, \frac{L}{2} + 1]$ , then its right upper corner  $A_{k+1}[x, y]$  is such that  $x < L + 1$  or  $y < L + 1$ . Notice that we may assume that either one tile will be formed by subarray  $A_{k+1}[x + 1, 1, n, y]$  or by subarray  $A_{k+1}[1, y + 1, x, n]$ . This is because the tile that contains  $A_{k+1}[x + 1, y + 1]$  cannot contain both  $A_{k+1}[1, y + 1]$  and  $A_{k+1}[x + 1, 1]$ . Suppose w.l.o.g. that the tile  $T_2$  that contains  $A_{k+1}[x + 1, 1]$  does not cover any cell of row  $y + 1$ . We may then extend the upper right corner of  $T_2$  till  $A_{k+1}[n, y]$ , without increasing the weight of any tile. We are left with  $2k$  tiles, and the induction hypothesis gives us the result.

In the case when  $p$  is odd, the input binary array  $A'_p$  is obtained from  $A_{k+1}$  by deleting any rows and columns in it with index at least  $k(L + 1) + \frac{L}{2} + 2$ . This, in effect adds an extra half  $L$ -cross near the upper right corner of  $A_k$ . Clearly, for  $p = 3$  it is not possible to tile  $A'_3$  with 3 tiles such that the weight of maximum weight tile is less than  $\frac{3}{2}L$ . The rest of the proof follows from arguments similar to the case when  $p$  is even. ◀

► **Remark.** The approximation factor of our algorithm is  $\frac{3}{2} + \frac{p^2}{w(A)}$  which is equal to  $(\frac{3}{2} + \frac{p}{L})L$ . Since  $\frac{p}{L}$  is equal to  $\frac{p^2}{w(A)}$ , it means that the approximation of our algorithm is tight under the condition that  $w(A) \gg p^2$ .

## 7 DRTILE

In this section, we present an approximation algorithm for the DRTILE problem. We have presented a  $\frac{3}{2} + \beta$ -approximation algorithm for the RTILE problem in Section 3. Now we show how to reduce an instance of the DRTILE problem to an instance of the RTILE problem to achieve an approximation ratio for the DRTILE problem. Before we proceed, let us recall the definition of the DRTILE problem.

### ■ the DRTILE problem

- **Input:** A two-dimensional array  $A$  and a weight upper bound  $w$ .
- **Goal:** Partition  $A$  into a minimum number of non-overlapping tiles, where the weight of each tile must not be larger than  $W$ .

Let us consider an array  $A$  with  $w(A) = n$ . Suppose  $W$ , provided as input, is the maximum allowed weight of any tile. Clearly, the minimal number of tiles we need to use to cover  $A$  is  $\lceil \frac{n}{W} \rceil$ . Consequently,  $\lceil \frac{n}{W} \rceil$  is a lower bound to the optimal solution of the DRTILE problem. Our goal is to obtain a  $\gamma$ -approximation algorithm, where  $\gamma$  depends on  $W$ . Therefore, the number of tiles we are allowed to use to cover  $A$  with this approximation is  $\gamma \times \lceil \frac{n}{W} \rceil$ .

We construct an instance of the the RTILE problem as follows: as an input we have the same array  $A$ , and we are allowed to use at most  $p = \gamma \times \lceil \frac{n}{W} \rceil$  tiles. Hence from Section 2, the lower bound on the maximum weight of a tile is  $\lceil \frac{n}{\gamma \times \lceil \frac{n}{W} \rceil} \rceil$ . Hence the maximum weight of a tile with approximation factor of  $\frac{3}{2} + \beta$  is,

$$\left\lceil \frac{n}{\gamma \times \lceil \frac{n}{W} \rceil} \right\rceil \times \left(\frac{3}{2} + \beta\right) \leq \left\lceil \frac{n}{\gamma \times \frac{n}{W}} \right\rceil \times \left(\frac{3}{2} + \beta\right) = \left\lceil \frac{W}{\gamma} \right\rceil \times \left(\frac{3}{2} + \beta\right).$$

## 28:16 Rectangle Tiling Binary Arrays

For the solution returned by RTILE to be a valid solution of DRTILE, the value of  $\lceil \frac{W}{\gamma} \rceil \times (\frac{3}{2} + \beta)$  must not exceed  $W$ . This allows us to derive a bound on the value of the approximation factor  $\gamma$ , we have,

$$\begin{aligned} \lceil \frac{W}{\gamma} \rceil \times (\frac{3}{2} + \beta) &\leq W \\ \Rightarrow \lceil \frac{W}{\gamma} \rceil &\leq \frac{W}{(\frac{3}{2} + \beta)} \\ \Rightarrow \frac{W}{\gamma} &\leq \frac{W}{(\frac{3}{2} + \beta)} + 1 \\ \Rightarrow \gamma &\geq (\frac{3}{2} + \beta) \cdot \frac{W}{W + (\frac{3}{2} + \beta)}. \end{aligned}$$

This gives us the following theorem.

► **Theorem 23.** *There exists a  $(\frac{3}{2} + \beta) \cdot \frac{W}{W + (\frac{3}{2} + \beta)}$ -approximation algorithm for the DRTILE problem where  $(\frac{3}{2} + \beta)$  is the approximation factor for the RTILE problem. The approximation factor of the DRTILE problem tends to  $\frac{3}{2}$  as the value of  $W$  is increased.*

## 8 The Multidimensional RTILE Problem

In Section 3, the algorithm presented for the RTILE problem was restricted to two dimensions. In this section, we generalize that algorithm for the  $d$ -dimensional RTILE problem, where  $d \geq 2$ . In the  $d$ -dimensional RTILE problem, we are given a  $d$ -dimensional array of size  $n$  in each dimension, containing 0/1 as entries, and we have to partition the array into  $p$  non-overlapping  $d$ -dimensional tiles such that the maximum weight of a tile in a tiling is minimized. Similarly to Section 3, we assume that  $\frac{p^d}{w(A)}$  is close to 0 and give a  $\frac{2d-1}{d}$ -approximation algorithm for the  $d$ -dimensional RTILE problem. Notice that the approximation ratio converges to 2 as we increase the value of  $d$ .

**Boundaries and Shadows.** The definition of the boundaries and their shadows is a generalization of the definitions in Section 3. The *type* of the boundaries in a  $d$ -dimensional array can be defined analogously. By  $[i]$ , we define the set of boundaries of dimension  $n \times n \times \dots \times 1 \times \dots \times n$ , where  $i^{th}$  dimension has size 1.

Let  $B_1, B_2, \dots, B_k \in [i]$ , we define  $T_i = \sum_{i=1}^k t(B_i)$ . Finally  $T$  is defined as  $\min\{T_1, T_2, \dots, T_d\}$ . The following lemma is analogous to Fact 2.

► **Lemma 24.** *Let  $T = \{T_1, T_2, \dots, T_d\}$ , then the array can be  $\frac{2d-1}{d}$ -tiled with  $T + 1$  tiles.*

We can estimate the minimal weight of the array using a linear program. The constraints of the linear program have a similar form as mentioned in Section 3. In two dimensional problem, each constraint is greater than  $1.5L$ . In the  $d$ -dimensional RTILE problem, each constraint is greater than  $\frac{2d-1}{d}L$ , instead of  $1.5L$ .

► **Lemma 25.** *Let  $T = \{T_1, T_2, \dots, T_d\}$ , then  $w(A) > TL$ .*

The proof of this lemma is analogous to Lemma 20.

► **Theorem 26.** *There exists a  $\frac{2d-1}{d}$ -approximation algorithm for the multi-dimensional RTILE problem assuming  $\frac{p^d}{w(A)}$  is negligible.*

---

**References**

---

- 1 Piotr Berman, Bhaskar DasGupta, S. Muthukrishnan, and Suneeta Ramaswami. Improved approximation algorithms for rectangle tiling and packing. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*, pages 427–436, 2001.
- 2 Piotr Berman and Sofya Raskhodnikova. Approximation algorithms for min-max generalization problems. *ACM Trans. Algorithms*, 11(1):5:1–5:23, 2014.
- 3 Moses Charikar, Chandra Chekuri, and Rajeev Motwani. Unpublished. Unpublished manuscript.
- 4 Wenliang Du, David Eppstein, Michael T. Goodrich, and George S. Lueker. On the approximability of geometric and geographic generalization and the min-max bin covering problem. In *Algorithms and Data Structures, 11th International Symposium, WADS*, pages 242–253, 2009.
- 5 Grzegorz Gluch and Krzysztof Lorys. 4/3 rectangle tiling lower bound. *CoRR*, abs/1703.01475, 2017. [arXiv:1703.01475](https://arxiv.org/abs/1703.01475).
- 6 Michelangelo Grigni and Fredrik Manne. On the complexity of the generalized block distribution. In *Parallel Algorithms for Irregularly Structured Problems, Third International Workshop, IRREGULAR '96*, pages 319–326, 1996.
- 7 Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson. On approximating rectangle tiling and packing. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 1998.
- 8 Sanjeev Khanna, S. Muthukrishnan, and Steven Skiena. Efficient array partitioning. In *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, pages 616–626, 1997.
- 9 Krzysztof Lorys and Katarzyna E. Paluch. Rectangle tiling. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX2000*, pages 206–213, 2000.
- 10 Krzysztof Lorys and Katarzyna E. Paluch. New approximation algorithm for RTILE problem. *Theor. Comput. Sci.*, 303(2-3):517–537, 2003.
- 11 S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In *Database Theory - ICDT '99*, pages 236–256, 1999.
- 12 Katarzyna Paluch. *Approximation Algorithms for Rectangle Tiling*. PhD thesis, University of Wrocław, Poland, 2004.
- 13 Katarzyna E. Paluch. A  $2(1/8)$ -approximation algorithm for rectangle tiling. In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004*, pages 1054–1065, 2004.
- 14 Katarzyna E. Paluch. A new approximation algorithm for multidimensional rectangle tiling. In *Algorithms and Computation, 17th International Symposium, ISAAC 2006*, pages 712–721, 2006.
- 15 Jonathan P Sharp. Tiling multi-dimensional arrays. In *International Symposium on Fundamentals of Computation Theory*, pages 500–511. Springer, 1999.
- 16 Adam Smith and Subhash Suri. Rectangular tiling in multidimensional arrays. *J. Algorithms*, 37(2):451–467, 2000.



# Approximation Algorithms for Correlated Knapsack Orienteering

David Alemán Espinosa ✉

Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1, Canada

Chaitanya Swamy ✉ 

Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1, Canada

---

## Abstract

---

We consider the *correlated knapsack orienteering* (CorrKO) problem: we are given a travel budget  $B$ , processing-time budget  $W$ , finite metric space  $(V, d)$  with root  $\rho \in V$ , where each vertex is associated with a job with possibly correlated random size and random reward that become known only when the job completes. Random variables are independent across different vertices. The goal is to compute a  $\rho$ -rooted path of length at most  $B$ , in a possibly adaptive fashion, that maximizes the reward collected from jobs that processed by time  $W$ . To our knowledge, CorrKO has not been considered before, though prior work has considered the uncorrelated problem, *stochastic knapsack orienteering*, and *correlated orienteering*, which features only one budget constraint on the *sum* of travel-time and processing-times.

Gupta et al. [19] showed that the *uncorrelated* version of this problem has a constant-factor adaptivity gap. We show that, perhaps surprisingly and in stark contrast to the uncorrelated problem, the *adaptivity gap of CorrKO is at least*  $\Omega(\max\{\sqrt{\log B}, \sqrt{\log \log W}\})$ . Complementing this result, we devise *non-adaptive* algorithms that obtain: (a)  $O(\log \log W)$ -approximation in quasi-polytime; and (b)  $O(\log W)$ -approximation in polytime. This also establishes that the adaptivity gap for CorrKO is at most  $O(\log \log W)$ . We obtain similar guarantees for CorrKO with cancellations, wherein a job can be cancelled before its completion time, foregoing its reward. We show that an  $\alpha$ -approximation for CorrKO implies an  $O(\alpha)$ -approximation for CorrKO with cancellations.

We also consider the special case of CorrKO where job sizes are weighted Bernoulli distributions, and more generally where the distributions are supported on at most two points (2CorrKO). Although weighted Bernoulli distributions suffice to yield an  $\Omega(\sqrt{\log \log B})$  adaptivity-gap lower bound for (uncorrelated) *stochastic orienteering*, we show that they are easy instances for CorrKO. We develop non-adaptive algorithms that achieve  $O(1)$ -approximation, in polytime for weighted Bernoulli distributions, and in  $(n + \log B)^{O(\log W)}$ -time for 2CorrKO. (Thus, our adaptivity-gap lower-bound example, which uses distributions of support-size 3, is tight in terms of support-size of the distributions.)

Finally, we leverage our techniques to provide a quasi-polynomial time  $O(\log \log B)$  approximation algorithm for correlated orienteering improving upon the approximation guarantee in [2].

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Mathematics of computing  $\rightarrow$  Discrete optimization

**Keywords and phrases** Approximation algorithms, Stochastic orienteering, Adaptivity gap, Vehicle routing problems, LP rounding algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.29

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2408.16566>

**Funding** *David Alemán Espinosa*: Supported in part by NSERC grant 327620-09.

*Chaitanya Swamy*: Supported in part by NSERC grant 327620-09.

## 1 Introduction

The *orienteering* problem, first introduced by [16], is a fundamental and widely-studied vehicle-routing problem (VRP). The input to the problem consists of a length/travel bound  $B$ , finite metric space  $(V, d)$  representing travel times, root vertex  $\rho \in V$ , and non-negative rewards



© David Alemán Espinosa and Chaitanya Swamy;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 29; pp. 29:1–29:24



Leibniz International Proceedings in Informatics

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

associated with the vertices. The goal is to compute a path rooted at  $\rho$  of length at most  $B$  that collects maximum reward. Orienteering often arises as a subroutine in devising algorithms for other more complex VRPs, both in approximation algorithms [4, 12, 6, 22, 13, 14, 1], as also in computational methods, where it arises as the pricing problem when using a branch-cut-and-price method on a set-covering/configuration LP.

Gupta et al. [19] introduced the following *stochastic* version of orienteering to model settings where one must spend some uncertain amount of time processing a visited node in order to collect its reward. Formally, each vertex corresponds to a job with a random, possibly correlated, processing time and reward, drawn from a given probability distribution. Random variables corresponding to different vertices are independent. The reward and processing time of a job become known only when the job is fully processed. The goal is to devise an algorithm, also called *policy*, that visits a sequence of vertices (starting at  $\rho$ ) in a possibly *adaptive* fashion that maximizes the expected total reward collected, subject to the constraint that the total time expended in traveling and processing jobs is at most  $B$ . Jobs cannot be preempted, and only jobs completed by the time-horizon  $B$  yield reward. This is the *correlated orienteering* (CorrO) problem. We refer to the special case where rewards and sizes are independent, simply as *stochastic orienteering*; due to independence, one can move to deterministic rewards by replacing the random rewards with their expectations.

A related problem, and the focus of this paper, is *correlated knapsack orienteering* (CorrKO), wherein there are two separate budgets:  $B$  for the (deterministic) travel time, and  $W$  for the total time spent in processing jobs. Again, we refer to the uncorrelated problem as *stochastic knapsack orienteering*. Correlated knapsack orienteering can be motivated from a similar perspective as CorrO. Indeed, it is quite natural to decouple the “apples and oranges” entities of travel time and processing time when these may represent disparate resources; e.g., travel time may represent latency of access of jobs in a distributed network, and processing time may present CPU time.

In general, a policy may be *adaptive* and choose the next vertex to visit based on the (size, reward) realizations of the vertices previously visited; unless otherwise stated, the approximation ratio is always measured relative to the maximum reward  $OPT$  that can be achieved by an adaptive policy. On the other hand, a *non-adaptive* policy fixes beforehand the sequence of vertices to visit, and only the stopping-point (when the time-horizon  $B$  is exceeded) depends on the (size, reward) realizations. While adaptive policies may collect much greater reward, non-adaptive policies are usually easier to implement, specify, and analyze, by virtue of the fact that they admit a much-more compact description compared to the decision tree associated with an adaptive policy whose description may require space that is exponential in the input size. Consequently, much work in stochastic optimization has focused on developing good non-adaptive policies and obtaining bounds on the *adaptivity gap*, which is the supremum, over all problem instances, of  $OPT/(\text{maximum reward achieved by a non-adaptive policy})$ ; see e.g., [9, 17, 11, 18, 19, 7, 20].

Prior work has studied stochastic orienteering, CorrO, and stochastic knapsack orienteering, and our current knowledge for these problems can be summarized as follows. (1) The adaptivity gap for stochastic orienteering is  $O(\log \log B)$  [19] and  $\Omega(\sqrt{\log \log B})$  [2], and there is a non-adaptive algorithm that achieves an  $O(1)$ -approximation with respect to the *non-adaptive optimum* [19], and hence obtains an  $O(\log \log B)$ -approximation; the approach leading to the latter result also yields an  $O(1)$ -approximation for *stochastic knapsack orienteering*. (2) The adaptivity gap for CorrO is also  $O(\log \log B)$  [2], but this is established non-constructively; algorithmically, we can obtain  $O(\alpha \log B)$ -approximation in polytime [19] and  $O(\alpha \cdot \frac{\log^2 \log B}{\log \log \log B})$  in quasi-polytime [2], where  $\alpha$  is the approximation ratio for *deadline-TSP*.



To our knowledge, *there is no prior work on CorrKO*. As noted above, (uncorrelated) stochastic knapsack orienteering and CorrO admit quite different guarantees, and this raises the natural question: where does CorrKO stand in terms of difficulty relative to these two problems? Is it more difficult than the uncorrelated problem? How does it compare in difficulty relative to CorrO?

## Our contributions

We initiate a study of correlated knapsack orienteering, and obtain results that, in particular, shed light on these questions. Our chief contributions are as follows.

1. **Adaptivity gap and approximation algorithms.** Somewhat surprisingly, and in stark contrast with (uncorrelated) stochastic knapsack orienteering, we prove that *the adaptivity gap for CorrKO is not a constant*, showing that the correlated problem is strictly harder than the uncorrelated problem.

► **Theorem 1.1** (see Section 3). *The adaptivity gap for CorrKO is  $\Omega(\max\{\sqrt{\log B}, \sqrt{\log \log W}\})$ , where  $B$  is the travel budget and  $W$  is the processing-time budget.*

Complementing this, we develop various *non-adaptive* approximation algorithms for CorrKO. Our main algorithmic result is a *quasi-polytime  $O(\log \log W)$ -approximation algorithm* for CorrKO, which thus shows that the adaptivity gap is  $O(\log \log W)$ .

► **Theorem 1.2.** *There are non-adaptive algorithms for CorrKO with the following guarantees:*

- (a)  $O(\log \log W)$ -approximation in time  $(n + \log B)^{O(\log W \log \log W)}$  (Section 4.1);
- (b)  $O(\log W)$ -approximation in polynomial time (Section 4.2).

By leveraging the approach leading to Theorem 1a, we also obtain the following guarantee for correlated orienteering, which improves upon the approximation guarantee in [2] (that also runs in quasi-polytime) by an  $O(\frac{\log \log B}{\log \log \log B})$ -factor.

► **Theorem 1.3.** *Given an  $\alpha$ -approximation algorithm for deadline TSP with running time  $T$ , we can obtain a non-adaptive  $O(\alpha \log \log B)$ -approximation algorithm for CorrO with running time  $(n + \log B)^{O(\log B \log \log B)} \cdot T$ . Using the algorithm for deadline TSP in [15], we obtain an  $O(\log \log B)$ -approximation in quasi-polytime.*

2. **CorrKO with 2-point distributions.** Our adaptivity-gap lower bound uses distributions of support-size 3, whereas the  $\Omega(\sqrt{\log \log B})$  adaptivity-gap lower-bound example for stochastic orienteering in [2] considers *weighted Bernoulli size distributions*. In Section 5, we investigate CorrKO from a fine-grained-complexity perspective to understand this discrepancy. In contrast with stochastic orienteering, we show that when all distributions are supported on at most 2 points – we call this 2CorrKO – *the adaptivity gap becomes  $O(1)$*  (Theorem 5.3), and we can obtain a *non-adaptive  $O(1)$ -approximation in time  $(n + \log B)^{O(\log W)}$*  (Theorem 5.4). Moreover, for weighted Bernoulli size distributions, we obtain a *polytime non-adaptive  $O(1)$ -approximation* (Theorem 5.5). Our key insight here lies in identifying a novel *deterministic* problem, that we call *orienteering with knapsack deadlines* (OrientKD), which we show is equivalent to 2CorrKO, up constant factors. In OrientKD, in addition to orienteering, each vertex  $v$  has a weight and knapsack deadline, and an orienteering-solution  $P$  is feasible, if for every  $v \in P$ , the total weight of all nodes on  $P$  up to (and including)  $v$  is at most its knapsack deadline. For instance, in a setting where jobs distributed over a network have to be processed on a single machine, travel times could represent the latency involved in accessing a job, and the knapsack deadlines would capture completion-time deadlines on the machine. We obtain

the above approximation guarantee for OrientKD (Theorem 5.1), and hence obtain the same guarantee (up to constant factors) for 2CorrKO.

These results show that our adaptivity-gap example is tight in terms of the support-size: *any* such lower-bound example *must* involve some distribution of support-size at least 3.

3. **CorrKO with cancellations.** In this version (see Section 6), we can *cancel* or discard the current vertex  $v$  at any time-step prior to its completion, foregoing its reward, and we are not allowed to return to  $v$ . We obtain the same approximation guarantees for this problem as for CorrKO: i.e., quasi-polytime  $O(\log \log W)$ -approximation, and polytime  $O(\log W)$ -approximation. En route, we obtain an  $O(1)$ -approximation for the special case where we obtain non-zero rewards only when jobs instantiate to size at most  $W/2$ .

Our results paint a nuanced picture of the complexity of CorrKO vis-a-vis CorrO and stochastic knapsack orienteering. While CorrKO is harder than stochastic knapsack orienteering, our algorithmic results suggest that it is easier than CorrO. We obtain similar approximation factors for both problems in quasi-polytime, but in polytime, we obtain  $O(\log W)$ -approximation for CorrKO, while the current-best polytime factor for CorrO is  $O(\log n \log B)$ ; also, with weighted Bernoulli distributions, CorrKO is provably easier than CorrO.

**Technical overview.** We now highlight the key technical ideas underlying our results. Let  $OPT$  be the optimal reward for CorrKO. Let  $S_v$  denote the random size of vertex  $v$ . For an integer  $j \geq 0$ , let  $X_v^j := \min\{S_v, 2^j\}$  and  $\mu_v^j = \mathbb{E}[X_v^j]$ . The significance of these quantities is that if  $\mu^j(P_{\rho,v} - v) \leq c \cdot 2^j$ , where  $P$  is a rooted path,  $v \in P$ , and  $P_{\rho,v}$  is the  $\rho \rightsquigarrow v$  portion of  $P$ , then a random subpath  $P''$  of  $P$  where we retain each  $u \in P$  independently with probability  $\frac{1}{2c}$  satisfies  $\Pr[v \in P'' \text{ and is processed by time } 2^j] \geq \frac{1}{4c}$ ; this indicates that  $\pi_v(2^j)$ , which is the expected reward of  $v$  if its processing starts by time  $2^j$ , can serve as a good proxy for the expected reward obtained from  $v$ .

**Algorithms for CorrKO and CorrO.** Our quasi-polytime  $O(\log \log W)$ -approximation for CorrKO builds upon a structural result for CorrO shown by [2]. They show that one can extract a suitable rooted path  $Q^*$  from the decision tree representing an optimal adaptive policy and suitable nodes  $\varphi_{-1} = \rho, \varphi_0, \varphi_1, \dots, \varphi_k$  on  $Q^*$ , where  $k \leq \log W$ , such that (roughly speaking): (a) the prefix property  $\mu^j(Q_{\rho, \varphi_j}^* - \varphi_j) \leq O(K) \cdot 2^j$  holds for every  $j = 0, \dots, k$ , and (b)  $\sum_{j=0}^k \sum_{v \in Q_{\varphi_{j-1}, \varphi_j}^*} \pi_v(2^j) = \Omega(OPT)$ , where  $K = O(\log \log W)$  (see Theorem 7.1). So if we could find this path  $Q^*$ , then using the sampling idea described above, one can easily obtain an  $O(K)$ -approximation. For CorrO, Bansal and Nagarajan [2] “guess” the *portal nodes*  $\varphi_0, \dots, \varphi_k$  and write a configuration LP to find suitable paths between every pair of consecutive portal nodes. They use randomized rounding to round a fractional solution, which incurs a  $\frac{\log k}{\log \log k}$ -factor violation of the prefix property due to Chernoff bounds, since for each  $j$ ,  $\mu^j(Q_{\rho, \varphi_j}^* - \varphi_j)$  can be written as a sum of  $O(K) \cdot 2^j$ -bounded independent random variables. When one combines this with the node-sampling step, one therefore incurs an  $O(K \cdot \frac{\log k}{\log \log k})$ -factor loss relative to the value of the LP solution.

For CorrKO (and CorrO), we proceed similarly, but we guess many more portal vertices. We split each  $Q_{\varphi_{j-1}, \varphi_j}^*$  into  $O(K)$  segments having  $\mu^j$ -weight at most  $2^j$ , and guess the end-points of all such segments (see Theorems 4.1, 4.2). We then again set up a configuration LP and use randomized rounding; however, we can now ensure that the prefix property holds with  $O(1)$  violation, since we can decompose  $\mu^j(Q_{\rho, \varphi_j}^* - \varphi_j)$  into a *sum of  $2^j$ -bounded random variables* corresponding to the  $\mu^j$ -weight of each random segment. Thus, an application of Chernoff bounds and the union bound only incurs an  $O(1)$ -factor violation of the prefix property,

since  $K = \Omega(\log k)$ ; therefore, we lose only an  $O(K)$ -factor compared to the value of the LP solution. This idea extends to **CorrO**. The only essential difference between **CorrKO** and **CorrO** comes from how well we can solve the corresponding configuration LP; for **CorrKO**, we can obtain an  $O(1)$ -approximation to the LP-optimum using an  $O(1)$ -approximation algorithm for knapsack orienteering (see below), but for **CorrO**, we obtain an  $O(\alpha)$ -approximate LP solution given an  $\alpha$ -approximation for deadline TSP.

The  $O(\log W)$ -approximation for **CorrKO** proceeds by relating the problem to *knapsack orienteering* (**KnapOrient**), which is orienteering with an additional total-node-weight budget constraint. For each index  $j = 0, 1, \dots, \log W$ , we use the portion of the optimal adaptive-policy tree corresponding to nodes processed at some point in  $[2^j, 2^{j+1})$ , to extract a good *fractional solution to an LP-relaxation* (**KO-LP**) for **KnapOrient**, where we exploit the LP-relaxation for orienteering [14]. This translation is easy because one can naturally interpret the LP variables as corresponding to certain probabilities obtained from an adaptive policy.

We remark that one can combine the LP-relaxations for orienteering [14] and the *correlated knapsack* problem [18], which is the special case where all nodes are co-located, to obtain an LP for **CorrKO**. However, the chief impediment in rounding an LP solution is that the rounding algorithms for orienteering and correlated knapsack may give rise to incompatible orderings. Rounding the orienteering-portion of the LP solution yields a node sequence, and we need to stick with a subsequence of this to satisfy the travel-budget constraint. However, forcing one to consider items in a prescribed order for correlated knapsack can drastically reduce the reward obtained, because jobs that instantiate to large sizes (i.e.,  $> W/2$ ) may need to be processed in a different incompatible order; see Appendix A. This tension is real, as evidenced by our adaptivity-gap lower bound, and seems challenging to deal with.

**2CorrKO.** The chief insight here is that the problematic case where we obtain reward only from large-size instantiations becomes quite structured in two ways. (1) There is no adaptivity gap, since only the path in the adaptive-policy tree corresponding to small-size (i.e.,  $\leq W/2$ ) instantiations can yield non-zero reward. (2) Given (1), one can infer that the reward obtained from a vertex  $v$  is a function of the total small size, and total large-size-instantiation probability of vertices visited up to  $v$ . This allows one to define an instance of *orienteering with knapsack deadlines* (**OrientKD**) to capture the stochastic problem.

**CorrKO with cancellations.** The algorithm for **CorrKO** with cancellations considers two cases. For large-size instantiations, it is not hard to argue that cancellations do not help (as with correlated knapsack [18]). For small-size instantiations, we formulate an LP by combining the LPs for orienteering [14] and *correlated knapsack with cancellations* [18]. We show that from an LP solution, one can define a suitable **KnapOrient**-instance and extract a good LP solution for this **KnapOrient**-instance. The **KnapOrient**-instance is defined in such a way that feasible solutions to this instance can be mapped to fractional solutions to the correlated-knapsack LP. So we can first round the solution to obtain an integral **KnapOrient**-solution  $Q$ , and then utilize the LP-rounding algorithm in [18] for correlated knapsack with cancellations to process vertices, with cancellations, *in the order they appear on  $Q$* . It is crucial here that the algorithm in [18] for small-size instantiations has the flexibility that one can specify a prescribed order for considering vertices (unlike in **CorrKO** with large-size instantiations).

## Related work

As mentioned earlier, *orienteering* is a fundamental problem in combinatorial optimization that finds various applications. Blum et al. [5] devised the first constant-factor approximation algorithm for orienteering, and the current best approximation factor is  $(2 + \epsilon)$  for any

$\epsilon > 0$  [8]. Friggstad and Swamy [14] gave the first LP-based  $O(1)$ -approximation algorithm. Their LP plays an important role for obtaining some of our results. *Deadline TSP*, also known as *deadline orienteering*, is a generalization of orienteering, where nodes now have deadlines, and a path  $P$  is feasible if, for every  $v \in P$ , its travel time along  $P$  is at most its deadline; the goal is again to compute a maximum-reward feasible path. Both orienteering and deadline TSP can be considered in the rooted, or *point-to-point* (P2P) setting, where both the start and end nodes of the path are specified. Deadline TSP admits a polytime  $O(\log n)$ -approximation [1] and an  $O(1)$ -approximation in time  $n^{O(\log(\text{maximum deadline}))}$  [15]. Friggstad and Swamy [15] also consider the more general *monotone-reward TSP*, wherein the reward of a node  $v$  having travel time  $t$  is given by  $\text{rewd}_v(t)$ , where  $\text{rewd}(\cdot)$  is a non-increasing function. They showed that this problem is essentially equivalent to deadline TSP.

The literature on stochastic optimization problems is rich, and we discuss below only the work that is most relevant to our work.

- **Stochastic knapsack problems.** Stochastic orienteering and CorrKO generalize respectively *stochastic knapsack*, which was studied in the seminal work of [9], and *correlated knapsack* [18, 21], which correspond to the special case where all nodes are co-located (i.e., the travel budget is irrelevant). The state-of-the-art for stochastic knapsack is a  $(2 + \epsilon)$ -approximation [3]. Gupta et al. [18] obtained the first constant-factor approximation for correlated knapsack, and the constant was improved to  $(2 + \epsilon)$  by Ma [21].
- **Stochastic VRPs.** We have already mentioned the works of Gupta et al. [19] and [2] that consider (uncorrelated) stochastic orienteering and correlated orienteering. A minimization version of stochastic orienteering, called *stochastic  $k$ -TSP* was considered by [11, 20], where instead of a travel budget, we want to collect a reward of at least  $k$ , and seek to minimize the expected travel time. Ene et al. [11] gave an adaptive  $O(\log k)$ -approximation algorithm for this problem, and Jiang et al. [20] obtained a non-adaptive  $O(1)$ -approximation. The special case where all nodes are co-located is called *stochastic knapsack cover* for which [10] obtained a  $(2 + \epsilon)$ -approximation.
- **Multi-armed bandits with metric switching costs.** A related problem to CorrKO is the *multi-armed bandit* problem with metric switching costs, considered by Guha and Munagala [17], which can be viewed as a setting where each vertex corresponds to a Markov chain (i.e., arm) with known transition probabilities and rewards. Guha and Munagala consider this setting under a crucial *martingale assumption*, which does not hold for CorrO or CorrKO, with separate budgets for the travel-cost and the number of arm-pulls, as in CorrKO. In their setting, one can also abandon a vertex and possibly return to this vertex at a later time. They devise an  $O(1)$ -approximation algorithm for this problem that is a hybrid between adaptive and non-adaptive policies: it non-adaptively specifies the sequence of arms to visit, but adaptively decides when an arm should be abandoned. They use an elegant Lagrangian-relaxation idea to reduce the problem to orienteering; this Lagrangian-relaxation idea was also later used in [19].

## 2 Preliminaries and notation

For an integer  $n \geq 0$ , we use  $[n]$  to denote  $\{1, \dots, n\}$ , where  $[0] := \emptyset$ , and  $\llbracket n \rrbracket$  to denote  $\{0\} \cup [n]$ . For any universe  $U$ , set  $S \subseteq U$  and element  $e \in U$ , we sometimes use  $S - e$  and  $S + e$  to denote  $S \setminus \{e\}$  and  $S \cup \{e\}$  respectively.

The problems we consider involve a metric space  $(V, d)$  and root  $\rho \in V$ . The metric  $d : V \times V \mapsto \mathbb{Z}_{\geq 0}$  is symmetric and captures travel times between vertices; by scaling we may assume that these are integers. Let  $n = |V|$  and  $\Delta$  be the diameter of the metric space. For

a set  $S$  of edges of the underlying complete graph  $(V, E)$ , we use  $d(S)$  to denote  $\sum_{e \in S} d(e)$ . Similarly, for any  $f \in \mathbb{R}^V$  and  $U \subseteq V$ ,  $f(U)$  denotes  $\sum_{v \in U} f_v$ . We say that a path  $P$  in  $G$  is rooted if it begins at  $\rho$ . We always think of the nodes on a rooted path  $P$  as being ordered in increasing order of their distance from  $\rho$  along the path. For any  $u, w \in P$ , we say  $u \prec_P w$  to denote that  $u$  comes before  $w$  on  $P$ , and  $u \preceq_P w$  means that  $u = w$  or  $u \prec_P w$ ; we omit the subscript  $P$  when  $P$  is clear from the context. We will interchangeably think of a path as an edge-set, or a sequence of nodes; the meaning will be clear from the context. For any path  $P$  and nodes  $a, b \in P$ , we use  $P_{a,b}$  to denote the  $a$ - $b$  portion of  $P$ . For a path  $P$  starting at node  $r$ , and a node  $v \in P$ , we define the travel time of  $v$  as  $d(P_{r,v})$ .

**Deterministic max-reward vehicle routing.** The following three vehicle routing problems (VRPs) play a prominent role in the study of stochastic orienteering. All three problems fall in the genre of max-reward VRPs, wherein we have nonnegative node rewards  $\{\pi_v\}_{v \in V}$ , and we need to select some vertices and find a suitable path visiting these vertices, so as to maximize the reward obtained. The differences in the problems lie in which paths are allowed, and the definition of the reward collected by a path. The problems below can be considered in the *rooted* setting, where we have a root  $\rho$  and the feasible paths form a subset of rooted paths, or in the *point-to-point* (P2P) setting, where both a start-node  $a$  and end-node  $b$  are specified, and the feasible paths are a subset of  $a$ - $b$  paths.

- **Orienteering.** We have a budget  $B$ , and feasible paths (in the rooted and P2P versions) are those with length at most  $B$ ; we collect the reward of all nodes on a feasible path.
- **Deadline TSP**, also called **deadline orienteering**. Here nodes have deadlines  $\{D_v\}_{v \in V}$ . A path  $P$  with the appropriate end-points is feasible if the travel time of each node in  $P$  is at most its deadline. So in the rooted case, a rooted path  $P$  is feasible if  $d(P_{\rho,v}) \leq D_v$  for all  $v \in P$ ; in the P2P-case, an  $a$ - $b$  path  $P$  is feasible if  $d(P_{a,v}) \leq D_v$  for all  $v \in P$ . We collect the reward of all nodes on a feasible path. (Equivalently, one can say that the feasible paths are *all* paths with the prescribed end-points, and we collect the reward from all nodes on the path that are visited *by their deadlines*.)

Observe that orienteering is the special case where the deadline of each node is the length bound  $B$ . Also, the rooted and P2P versions of deadline TSP are equivalent [14].

- **Monotone-reward TSP.** This is a generalization of deadline TSP, where each node  $v$  has a non-increasing reward-function  $\pi_v : \mathbb{Z}_+ \mapsto \mathbb{R}_+$ , where  $\pi_v(t)$  gives the reward obtained from  $v$  if  $v$  is visited at time  $t$ . Every path  $P$  with the appropriate end-points is feasible, and the reward of  $P$  is given by  $\sum_{v \in P} \pi_v(\text{travel time of } v) = \sum_{v \in P} \pi_v(d(P_{r,v}))$ , where  $r$  is the start node of  $P$ .

Friggstad and Swamy [14] showed that monotone-reward TSP can be reduced to deadline TSP losing a  $(1 + \epsilon)$ -factor, for an  $\epsilon > 0$ . Monotone-reward TSP will play a key role in the algorithm for correlated orienteering.

**Stochastic orienteering problems.** In the *correlated knapsack orienteering* (CorrKO) problem, each vertex  $v \in V$  is associated with an stochastic job with a random processing time or size  $S_v \in \mathbb{Z}_{\geq 0}$  and a possibly *correlated* random reward  $R_v \in \mathbb{R}_{\geq 0}$ . We use the terms processing time and size interchangeably. These random variables are independent across different vertices, and their distributions are specified in the input. We are given a length or travel-time budget  $B$ , and a processing-time budget  $W$ . A solution, or policy, for CorrKO visits a sequence of (distinct) vertices starting from the root  $\rho$ , in a possibly adaptive fashion, without exceeding the travel-time and processing-time budgets. More precisely, when a vertex  $v$  is visited, it's corresponding job is processed non-preemptively, and we get to know

the processing time and reward of the job only upon its completion; the completion time of job  $v$  is the total processing time of all jobs up to and including  $v$ . So if the adaptive policy visits vertices  $v_0 := \rho, \dots, v_\ell = u$  in that order, then it must be that the total travel-time  $\sum_{i=1}^{\ell} d(v_{i-1}, v_i)$  to get to  $u$  is at most  $B$ , and the total processing time of (the jobs associated with)  $v_1, \dots, v_{\ell-1}$  is at most  $W$ . We collect the rewards of  $v_1, \dots, v_{\ell-1}$ , and we collect  $u$ 's reward if its completion time is at most  $W$ . The goal is to maximize the expected total reward collected. For notational convenience, we also assign a deterministic value of 0 to the reward and processing time of  $\rho$ .

In the *correlated orienteering* (CorrO) problem, the setup is almost the same as in CorrKO, except that there is only one budget  $B$ , which is the budget for the *sum* of the travel times and processing times. (That is, we have one notion of time, which advances due to both travel and the processing of jobs.) So if an adaptive policy for CorrO visits vertices  $v_0 := \rho, \dots, v_\ell = u$  in that order, then we must have  $\sum_{i=1}^{\ell} d(v_{i-1}, v_i) + \sum_{i=1}^{\ell-1} S_{v_i} \leq B$ ; that is, the completion time of each  $v_i$  for  $i = 1, \dots, \ell - 1$ , as also the time when we reach  $v_\ell$ , *taking into account both travel time and processing time*, should be at most  $B$ . We collect rewards from  $v_1 \dots, v_{\ell-1}$ , and we collect  $u$ 's reward if  $\sum_{i=1}^{\ell} d(v_{i-1}, v_i) + \sum_{i=1}^{\ell-1} S_{v_i} \leq B$ .

Any adaptive policy for CorrKO or CorrO can be represented by a decision tree  $\mathcal{T}$  rooted at  $\rho$ , whose nodes are labeled by vertices of  $V$ , and the branches of a node labeled  $v \in V$  correspond to the different size and reward instantiations of  $v$ , with each branch specifying the next node to visit under the corresponding instantiation.

A *nonadaptive policy* (for CorrKO or CorrO) fixes a priori the sequence of vertices to potentially visit, *without looking at the size and reward instantiations*. The *adaptivity gap* for an instance is the ratio (optimal expected reward collected by an adaptive policy)/(optimal reward collected by a nonadaptive policy), and the adaptivity gap for a problem is the supremum over all instances of the adaptivity gap for the instance.

**Deterministic knapsack-constrained vehicle routing.** Algorithms for stochastic orienteering problems frequently utilize knapsack-constrained variants of deterministic VRPs, wherein we seek a feasible solution to the VRP satisfying an additional knapsack constraint on the total vertex-weight of the path. More precisely, suppose we have an underlying “base” max-reward VRP, specified by a collection  $\mathcal{I}$  of feasible paths along with nonnegative vertex-rewards  $\{\pi_v\}_{v \in V}$ , where the goal is to find a maximum-reward path in  $\mathcal{I}$ . In the *knapsack-constrained version of this VRP*, we also have a knapsack constraint specified by nonnegative knapsack weights  $\{\text{wt}_v\}_{v \in V}$  and knapsack budget  $W$ , which restricts the set of feasible solutions to  $\mathcal{I}^{\text{knap}} := \{\tau \in \mathcal{I} : \sum_{v \in \tau} \text{wt}_v \leq W\}$ ; the goal is to find a maximum-reward path in  $\mathcal{I}^{\text{knap}}$ , i.e., a maximum-reward path in  $\mathcal{I}$  satisfying the knapsack constraint. When the base VRP is: (i) orienteering, the knapsack-constrained version is *knapsack orienteering* (KnapOrient); (ii) deadline-TSP, the knapsack-constrained version is *knapsack deadline orienteering* (KnapDO). KnapOrient and KnapDO were considered by [19, 2] in the context of stochastic orienteering. We say that the base-VRP is a rooted-VRP, if all paths in  $\mathcal{I}$  start at the same vertex, and it is a P2P-VRP, if all paths in  $\mathcal{I}$  have the same start and end nodes.

We give a general reduction (Theorem 2.1) showing if the base-VRP is a rooted-VRP or P2P-VRP, and satisfies a certain subpath-closure property, then an  $\alpha$ -approximation for the VRP can be used as a black-box to obtain an  $(\alpha + 2)$ -approximation for the knapsack-constrained VRP. Let  $\tau$  be a path with ends  $a, b \in V$ , which we will view as a sequence of nodes. By a P2P-*subpath* of  $\tau$ , we mean any  $a$ - $b$  path whose node-sequence is a subsequence of  $\tau$ ; by a *rooted-subpath* of  $\tau$ , we mean any path starting at  $a$  whose node-sequence is a subsequence of  $\tau$ . (Note that any subsequence of  $\tau$  yields a path, since we are working with

a complete graph.) The *subpath-closure property* requires that for every path  $\tau \in \mathcal{I}$ : (a) for rooted-VRP, every rooted-subpath  $\tau'$  of  $\tau$  is also in  $\mathcal{I}$ , (b) for P2P-VRP, every P2P-subpath  $\tau'$  of  $\tau$  is also in  $\mathcal{I}$ . Most max-reward VRPs – e.g., orienteering, deadline TSP – satisfy the subpath-closure property. (Also, note that if a VRP satisfies the subpath-closure property, then so does the knapsack-constrained VRP.)

The above reduction is based on a Lagrangian-relaxation idea that was also used by [19], specifically to obtain approximation algorithms for **KnapOrient** and **KnapDO**. However, their approach results in a constant-factor blowup -in the approximation ratio (factor 2 for **KnapOrient**, and factor 4 for **KnapDO**<sup>1</sup> when going from the VRP to the knapsack-constrained VRP; our general reduction yields a better factor, in a somewhat simpler fashion.

► **Theorem 2.1.** *Consider a max-reward rooted-VRP or P2P-VRP, specified by a set  $\mathcal{I}$  of feasible solutions satisfying the subpath-closure property. For any  $\epsilon > 0$ , an  $\alpha$ -approximation algorithm  $\mathcal{A}$  (where  $\alpha \geq 1$ ) for the VRP can be used to obtain an  $(\alpha + 2)(1 + \epsilon)$ -approximation for the knapsack-constrained VRP by making  $O(\frac{\log n}{\epsilon})$  calls to  $\mathcal{A}$ .*

► **Corollary 2.2.** *There are algorithms with the following guarantees.*

- (a)  $(4 + \epsilon)$ -approximation, for any  $\epsilon > 0$ , for rooted- and P2P- knapsack orienteering;
- (b)  $O(\log n)$ -approximation for the rooted and P2P versions of knapsack deadline orienteering, and knapsack monotone-reward TSP;
- (c)  $O(1)$ -approximation in  $O(n^{\log n \Delta})$  time, for the rooted and P2P versions of knapsack deadline orienteering, and knapsack monotone-reward TSP.

**LP-relative guarantee for **KnapOrient**.** For rooted **KnapOrient**, we can utilize Theorem 2.1 to obtain an LP-relative approximation guarantee. This will be useful in devising algorithms for **CorrKO**. Consider the following LP-relaxation for rooted **KnapOrient** along the lines of an LP-relaxation for rooted orienteering in [14]. Let  $\rho$  be the root node for the **KnapOrient** instance. We bidirect the edges of the complete graph on  $V$  to obtain the arc-set  $A$ .

$$\max \quad \sum_{u,v \in V} z_u^v \cdot \pi_u \quad (\text{KO-LP})$$

$$\text{s.t.} \quad x^v(\delta^{\text{in}}(u)) \geq x^v(\delta^{\text{out}}(u)) \quad \forall u \in V - \rho, v \in V \quad (\text{O1})$$

$$x^v(\delta^{\text{in}}(S)) \geq z_u^v \quad \forall v \in V, S \subseteq V - \rho, u \in S \quad (\text{O2})$$

$$z_u^v = 0 \quad \forall u, v \in V : d_{\rho,u} > d_{\rho,v} \quad (\text{O3})$$

$$\sum_{a \in A} d_a \cdot x_a^v \leq Bz_v^v, \quad x^v(\delta^{\text{out}}(\rho)) = z_v^v \quad \forall v \in V \quad (\text{O4})$$

$$x, z \geq 0$$

$$\sum_{v \in V} z_v^v = 1 \quad (\text{O5}) \quad \sum_{u,v \in V} z_u^v \cdot \text{wt}_u \leq W. \quad (\text{KN})$$

The  $x_a^v$  and  $z_u^v$  variables encode the arcs included, and vertices visited, respectively by the **KnapOrient**-path, provided that  $v$  is the node visited that is furthest from  $\rho$ , i.e.,  $v$  maximizes  $d(\rho, u)$  among all nodes  $u$  on the path: constraints (O3) enforce this semantics; in an integer solution, these variables will be 0 if  $v$  is not the furthest visited node from  $\rho$ . Constraints

<sup>1</sup> [19] do not explicitly state a result for **KnapDO**, and instead embed this result within their algorithm for correlated orienteering. We can infer this factor by tracing through their algorithm and analysis.

(O1) and (O2) encode that the  $\rho \rightsquigarrow u$ -connectivity is  $z_u^v$ , and together with (O4) encode that  $\{x_a^v\}$  is a  $\rho$ -preflow of value  $z_v^v$  satisfying the length budget. Constraint (O5) enforces that overall  $x$  is a  $\rho$ -preflow of value 1. Constraints (O1)–(O5) are from the LP for rooted orienteering in [14]; (KN) is the new constraint encoding the knapsack budget.

► **Theorem 2.3.** *We can obtain a KnapOrient-solution that obtains reward at least  $OPT_{KO-LP}/5$ .*

### 3 An adaptivity-gap lower bound for CorrKO

We now show that the adaptivity gap for CorrKO is  $\Omega(\max\{\sqrt{\log B}, \sqrt{\log \log W}\})$ , thereby proving Theorem 1.1. We consider the following instance of correlated knapsack orienteering that has a similar spirit as the adaptivity-gap example in [2] for (uncorrelated) stochastic orienteering. The metric is a tree-metric induced by a complete binary tree  $T$  on a vertex set  $V$ , with root  $r \in V$  and  $H \geq 4$  levels, where the distances decrease geometrically as we move away from  $r$ . To conform to our notation, we include a separate dummy node  $\rho$  that serves as the root for CorrKO, with distance 0 to  $r$ ; but when we say root below, we always mean the root  $r$  of the tree  $T$ . The knapsack budget is  $W := 2^{2^{H+1}}$  and the length/travel budget is  $B := 2^{H-1} - 1$ . For a node  $v \in V$ , we use:  $\text{lev}(v)$  to denote the level of  $v$ ,  $\text{path}(v)$  to denote the unique  $r \rightsquigarrow v$ -path in  $T$ , and  $\text{par}(v)$  to denote the parent of  $v$  if  $v \neq r$ . The root  $r$  is at level  $H$  and each leaf node is at level 1; for a non-leaf node  $v$  at level  $\ell$ , the distance between  $v$  and its children is  $2^{\ell-2}$ . For a rooted path  $P$  in  $T$  we say that a node  $v \in P$  is a right-branching (resp. left-branching) node if the node succeeding  $v$  on  $P$  is its right-child (resp. left-child). We denote by  $\text{rt}(P)$  and  $\text{left}(P)$ , the right-branching nodes and left-branching nodes of  $P$ , respectively. For notational convenience, we assume that the end-node of  $P$  other than  $r$  is a left-branching node; if  $P = r$ , then we say that  $r \in \text{left}(P)$ . The (correlated) (size, reward) distribution of node  $v$  is supported on three points:

$$\begin{aligned} (S_v^{(3)}, R_v^{(3)}) &= (0, 0), & (S_v^{(2)}, R_v^{(2)}) &= \left( 2^{2^{\text{lev}(v)}} \cdot \prod_{w \in \text{rt}(\text{path}(v))} 2^{2^{\text{lev}(w)}}, 0 \right) \\ (S_v^{(1)}, R_v^{(1)}) &= \left( W - \sum_{w \in \text{rt}(\text{path}(v))} S_w^{(2)}, \left( 1 - \frac{1}{\sqrt{H}} \right)^{|\text{rt}(\text{path}(v))|} \right) \end{aligned}$$

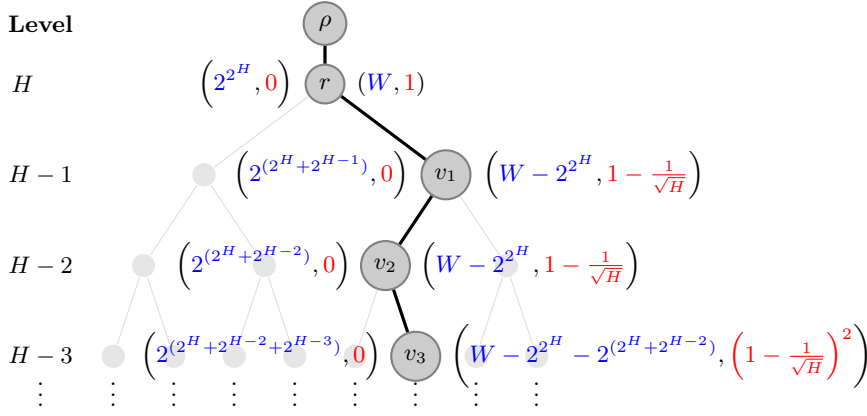
and we have  $\Pr[S_v = S_v^{(3)}] = 1 - \frac{1}{\sqrt{H}} - \frac{1}{H}$ ,  $\Pr[S_v = S_v^{(2)}] = \frac{1}{\sqrt{H}}$ ,  $\Pr[S_v = S_v^{(1)}] = \frac{1}{H}$ ; see Fig. 1. Observe that  $S_v^{(2)} \leq (2^{2^{\text{lev}(v)+1}})/2 \leq W/2$ , and  $S_v^{(1)} > W/2$  for every node  $v$ .

Importantly, note that any policy for this instance can obtain positive reward from at most one item. This is because for any  $v \in V$ ,  $S_v^{(1)} > W/2$ . Therefore we can assume that any policy terminates upon observing a size  $S_v^{(1)}$  for any visited vertex  $v$ . The binary tree is built so that a certain adaptive policy (see the proof of Theorem 3.1) can always reach a leaf-node if no positive reward has been collected in previous levels. The construction of the tree prevents any path from going upward from a node to its parent, as this will cause the length budget to run out. But more importantly, the instance is set up to preclude a policy from going to a left child of a node  $v$  if its instantiated size is  $S_v^{(2)}$  in the sense that if this happens then one cannot collect positive reward from this point on (Lemma 3.4).

The adaptivity-gap lower bound immediately follows from Theorems 3.1 and 3.2, since  $H = \Omega(\log B)$  and  $H = \Omega(\log \log W)$  for the above CorrKO instance.

► **Theorem 3.1.** *There is an adaptive policy for the above CorrKO instance that obtains  $\Omega(1)$  expected reward.*





■ **Figure 1** The  $(S_v^{(2)}, R_v^{(2)})$ ,  $(S_v^{(1)}, R_v^{(1)})$  pairs are shown respectively on the left and right of each highlighted vertex in the tree.

► **Theorem 3.2.** *Any nonadaptive policy for the above CorrKO instance obtains expected reward at most  $\frac{2}{\sqrt{H}}$ .*

**Proof of Theorem 3.1.** Consider the following adaptive policy  $\mathcal{A}$ : the policy moves to node  $r$  from  $\rho$ , and then proceeds as follows. Let  $v$  be the current node visited, which is  $r$  initially. If  $v$  is a leaf, then the policy ends after the instantiation of  $v$ . Otherwise, the next node visited by  $\mathcal{A}$  is: the left child of  $v$ , if  $S_v = S_v^{(3)}$ , and the right child of  $v$  if  $S_v = S_v^{(2)}$ ; if  $S_v = S_v^{(1)}$ , then  $\mathcal{A}$  stops and does not visit any other nodes.

Let  $P^*$  denote the (random) path traversed by  $\mathcal{A}$ , which we may view as a rooted path in  $T$ . Let  $v_{\text{last}}$  be the last vertex visited by  $\mathcal{A}$ , i.e.,  $v_{\text{last}}$  is the end-node of  $P^*$  other than  $r$  and  $P^* = \text{path}(v_{\text{last}})$ .

▷ **Claim 3.3.** We have  $d(P^*) \leq B$  and  $S(P^*) := \sum_{v \in P^*} S_v \leq W$  with probability 1.

We argue that the expected reward collected by  $P^*$  is at least  $\frac{1-e^{-1}}{4}$ . Let  $\mathcal{R} = R(P^*) := \sum_{v \in P^*} R_v$  denote the reward obtained by  $P^*$ . Note that  $v_{\text{last}}$  is the only vertex from which  $P^*$  can collect positive reward. So  $\mathbb{E}[\mathcal{R}] = \Pr[S_{v_{\text{last}}} = S_{v_{\text{last}}}^{(1)}] \cdot \mathbb{E}[R_{v_{\text{last}}}^{(1)}]$ . Observe that if  $S_{v_{\text{last}}} \neq S_{v_{\text{last}}}^{(1)}$ , then  $v_{\text{last}}$  is a leaf node, and hence the event  $\{S_{v_{\text{last}}} \neq S_{v_{\text{last}}}^{(1)}\}$  occurs precisely when  $\mathcal{A}$  visits  $H$  vertices, one on each level of  $\mathcal{T}$ , and none of them instantiate to size  $S_v^{(1)}$ . Since vertex sizes are independent across different vertices, we have  $\Pr[S_{v_{\text{last}}} \neq S_{v_{\text{last}}}^{(1)}] = (1 - \frac{1}{H})^H \leq e^{-1}$ , and so  $\mathbb{E}[\mathcal{R}] \geq (1 - e^{-1})\mathbb{E}[R_{v_{\text{last}}}^{(1)}]$ .

We have  $R_{v_{\text{last}}}^{(1)} = (1 - \frac{1}{\sqrt{H}})^{|\text{rt}(P^*)|}$ , and since  $(1 - \frac{1}{\sqrt{H}})^x$  is a convex function, we obtain that  $\mathbb{E}[R_{v_{\text{last}}}^{(1)}] \geq (1 - \frac{1}{\sqrt{H}})^{\mathbb{E}[|\text{rt}(P^*)|]}$ . Observe that  $v \in P^*$  gets included in  $\text{rt}(P^*)$  precisely when  $S_v$  instantiates to  $S_v^{(2)}$ , which happens with probability  $\frac{1}{\sqrt{H}}$ . So we can upper bound  $\mathbb{E}[|\text{rt}(P^*)|]$  by  $H \cdot \frac{1}{\sqrt{H}} = \sqrt{H}$ . It follows that  $\mathbb{E}[R_{v_{\text{last}}}^{(1)}] \geq (1 - \frac{1}{\sqrt{H}})^{\sqrt{H}} \geq \frac{1}{4}$ , where the last inequality uses the fact that  $1 - x \geq 4^{-x}$  for  $x \leq 0.5$ . ◀

**Proof of Theorem 3.2.** Let  $\sigma$  be some non-adaptive policy, which we may again view as a rooted path in  $T$ , since we can always move first to  $r$ . We may assume that  $\sigma$  visits vertices in decreasing order of their levels, since any backtracking from a node  $v$  to its ancestor would cause one to exceed the travel budget. We say that an execution of  $\sigma$  “cheats” if, for some visited node  $v$ ,  $S_v$  instantiates to  $S_v^{(2)}$ , and  $\sigma$  proceeds to visit a vertex in the subtree of  $T$  rooted at the left-child of  $v$ .

► **Lemma 3.4.**  $\sigma$  does not collect any positive reward after cheating.

**Proof.** Suppose  $\sigma$  cheats at some vertex  $u$ . Let  $v$  be any node in the tree rooted at the left-child of  $u$ . The residual knapsack budget after visiting  $u$  is at most  $W - S_u^{(2)}$ . It suffices to show that  $S_v^{(1)} > W - S_u^{(2)}$ . Since  $S_v^{(1)} = W - \sum_{w \in \text{rt}(\text{path}(v))} S_w^{(2)}$ , this amounts to showing that  $S_u^{(2)} > \sum_{w \in A} S_w^{(2)}$ , where  $A = \text{rt}(\text{path}(v))$ . We argue that  $S_w^{(2)} < S_u^{(2)}$  for every  $w \in A$ , and the  $S_w^{(2)}$ 's are distinct for  $w \in A$ . This, coupled with the fact that  $S_u^{(2)}$  and the  $S_w^{(2)}$ 's are all powers of 2, implies the above inequality.

Recall that for a node  $z$ , we have  $S_z^{(2)} = 2^{2^{\text{lev}(z)}} \cdot \prod_{w \in \text{rt}(\text{path}(z))} 2^{2^{\text{lev}(w)}}$ . Let  $A = \{a_1, a_2, \dots, a_{|A|}\}$ , where the nodes are ordered in increasing order of their distance from  $r$ . Then, for any  $i \geq 2$ , we have  $S_{a_i}^{(2)} = 2^{2^{\text{lev}(a_i)}} \cdot S_{a_{i-1}}^{(2)}$ , showing that each  $S_{a_i}^{(2)}$  is a distinct power of 2, and  $S_{a_i}^{(2)}$  increases with  $i$ . Note that  $u \notin \text{rt}(\text{path}(v))$  and  $\text{rt}(\text{path}(u)) \subseteq A$ . So for  $z = a_{|A|}$ , we have  $S_z^{(2)} = \prod_{w \in A - \text{rt}(\text{path}(u))} 2^{2^{\text{lev}(w)}} \cdot \prod_{w \in \text{rt}(\text{path}(u))} 2^{2^{\text{lev}(w)}}$  and  $\prod_{w \in A - \text{rt}(\text{path}(u))} 2^{2^{\text{lev}(w)}} < 2^{2^{\text{lev}(u)}}$ . It follows that  $S_z^{(2)} < S_u^{(2)}$ . ◀

Recall that we view  $\sigma$  also as a rooted path in  $T$ . We can show that the total expected reward obtained from  $\text{rt}(\sigma)$  and  $\text{left}(\sigma)$  are both at most  $\frac{1}{\sqrt{H}}$ , which completes the proof. For the latter bound, we utilize the fact that, due to Lemma 3.4, we can collect positive reward from a node  $v$  only if  $S_w = S_w^{(3)}$  for every  $w \in \text{left}(\text{path}(v)) - v$ . ◀

## 4 Approximation algorithms for CorrKO

We now devise non-adaptive approximation algorithms for CorrKO. In Section 4.1, we develop an  $O(\log \log W)$ -approximation algorithm with  $(n + \log B)^{O(\log W \log \log W)}$  (i.e., quasi-polynomial) running time, which will prove Theorem 1a, and in Section 4.2, we obtain a polytime  $O(\log W)$ -approximation algorithm, thereby proving Theorem 1b.

### 4.1 Quasi-polytime $O(\log \log W)$ -approximation algorithm

There are two chief components underlying our algorithm. First, we isolate a key structural result (Theorems 4.1 and 4.2) showing that from an optimal adaptive policy, one can extract a suitable path  $Q^*$  and certain “portal” vertices on this path, such that the subpaths of  $Q^*$  between these portal vertices satisfy various nice properties. Second, we exploit this structural result algorithmically as follows. The structural result allows us to reduce the problem, at the expense of an  $O(\log \log W)$ -factor loss, to that of finding the portal vertices, and suitable paths between these portal vertices that satisfy certain knapsack constraints on the total expected truncated size  $\mathbb{E}[\min\{S_v, 2^j\}]$  of nodes on these paths. We “guess” (i.e., enumerate over all possible choices of) these portal vertices and some auxiliary information, and set up a configuration LP (CKO-P) to find paths between these portal vertices. This configuration LP can be solved near-optimally, and we show that a fractional solution can be rounded incurring only an  $O(1)$ -factor loss in the objective and in the constraints. Finally, we argue that this leads to an  $O(\log \log W)$ -approximation non-adaptive policy.

Our approach is similar in spirit to the one in [2] for CorrO, and in Section 7, we show that our approach also yields an  $O(\log \log B)$ -approximation for CorrO, which improves upon the guarantee in [2] by an  $O(\frac{\log \log B}{\log \log \log B})$ -factor. While we borrow various ingredients from [2], the key difference between our approach and theirs is that we extract much more information from the adaptive policy in terms of so-called portal vertices, which enables us to round an underlying configuration LP incurring only a *constant-factor* violation in the knapsack constraints; in contrast, this step in [2] incurs an  $O(\frac{\log \log B}{\log \log \log B})$ -factor violation of the constraints, and this savings is the source of our improved guarantee.

**Structural results.** Recall that, for a path  $P$  and nodes  $a, b \in P$ , we use  $P_{a,b}$  to denote the  $a$ - $b$  portion of  $P$ . If  $P$  is a  $u$ - $v$  path, its *regret* is  $d^{\text{reg}}(P) := d(P) - d(u, v)$ , and the *two-point regret* of  $P$  with respect to a node  $a \in P$  is  $d^{\text{reg}}(P, a) := d(P) - d(u, a) - d(a, v) = d^{\text{reg}}(P_{u,a}) + d^{\text{reg}}(P_{a,v})$ . For an index  $j \in \{0, 1, \dots, L := \lceil \log W \rceil\}$ , recall that we define  $X_v^j := \min\{S_v, 2^j\}$  and  $\mu_v^j := \mathbb{E}[X_v^j]$ . For any vertex  $v \in V$ , let  $\pi_v(t) := \mathbb{E}[R_v \cdot \mathbb{1}_{S_v \leq W-t}] = \sum_{t'=0}^{W-t} \Pr[S_v = t'] \cdot \mathbb{E}[R_v | S_v = t']$  denote the expected reward obtained from  $v$  if its processing starts at time  $t$ . Note that  $\pi_v(t) = 0$  for any  $t > W$ . Also, note that  $\pi_\rho(t) = 0$  for all  $t$ . We may assume that  $\pi_v(0) \leq OPT/4$  for every  $v \in V$ , as otherwise, we can obtain  $\Omega(OPT)$  reward by going to a single node.

Throughout, let  $K = 3 \log(6 \log W) + 12$ ,  $L = \lceil \log W \rceil$ ,  $N_1 = 2(K + 1)$ . Define  $\varphi_{-1} := \rho$ .

► **Theorem 4.1.** *There exists a rooted path  $Q^*$  with  $d(Q^*) \leq B$ , vertices  $\varphi_0 \preceq \varphi_1 \preceq \dots \preceq \varphi_k$  on  $Q^*$  for some  $k \leq L$ , and, for each  $j \in \llbracket k \rrbracket$ , a vertex-set  $\text{Por}_j \subseteq Q_{\varphi_{j-1}, \varphi_j}^*$  containing nodes  $\varphi_{j-1}, \varphi_j$ , with  $|\text{Por}_j| \leq N_1$ , whose vertices are ordered by the order they appear on  $Q^*$ , satisfying the following properties.*

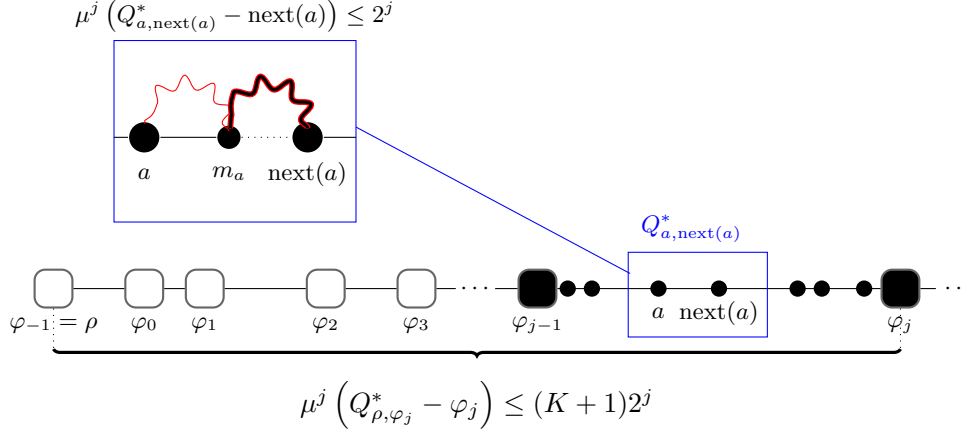
- (a)  $\sum_{j=0}^k \sum_{v \in Q_{\varphi_{j-1}, \varphi_j}^* - \varphi_j} \pi_v(2^j - 1) \geq OPT/4$ .
- (b)  $\mu^j(Q_{\rho, \varphi_j}^* - \varphi_j) \leq (K + 1)2^j$  for all  $j \in \llbracket k \rrbracket$ .
- (c) For every  $j \in \llbracket k \rrbracket$  and consecutive nodes  $a, b \in \text{Por}_j$ , we have  $\mu^j(Q_{a,b}^* - b) \leq 2^j$ .

As mentioned earlier, in our quasi-polytime algorithm, we utilize Theorem 4.1 to construct a good rooted path, by using enumeration to guess  $\bigcup_j \text{Por}_j$ , and an LP to then obtain suitable paths between consecutive nodes of  $\bigcup_j \text{Por}_j$ . In order to ensure that the total path length is at most the travel budget  $B$ , we will also need to obtain some information about the lengths  $d(Q_{a,b}^*)$  for consecutive nodes  $a, b$  in  $\bigcup_j \text{Por}_j$ . Naively guessing these lengths would yield incur a large  $B^{O(LN_1)}$ -factor in the running time; to do better, and reduce the dependence to  $(\log B)^{O(LN_1)}$ , we instead guess the two-point regret of each  $Q_{a,b}^*$  with respect to a “mid-point” node, within a factor of 2, which suffices (see Fig. 2). We refine Theorem 4.1 to incorporate these estimates as follows.

► **Theorem 4.2 (Main structural result).** *Let the node-sequence  $\varphi_0, \dots, \varphi_k$ , where  $k \leq L$ , and for each  $j \in \llbracket k \rrbracket$ , the ordered node sequence  $\text{Por}_j$  of at most  $N_1$  nodes, be as given by Theorem 4.1. Define  $\text{Por} := \bigcup_{j=0}^k \text{Por}_j$ , which we call “portal nodes”, where the ordering of nodes in  $\text{Por}$  is  $\text{Por}_0, \text{Por}_1, \dots, \text{Por}_k$ ; for  $a \in \text{Por}$ ,  $a \neq \varphi_k$ , let  $\text{next}(a)$  be the next node in  $\text{Por}$  after  $a$ . For each  $a \in \text{Por} - \varphi_k$ , there exists an  $a$ - $\text{next}(a)$  path  $Q_{a, \text{next}(a)}^*$ , auxiliary node  $m_a$ , and integer  $\gamma_a \geq 0$ , such that the following properties hold.*

- (P1) (Distance)  $d(Q_{a,b}^*) \leq D_a := 2^{\gamma_a} - 1 + d(a, m_a) + d(m_a, b)$  for every pair of consecutive nodes  $a, b \in \text{Por}$ .
- (P2) (Total-length)  $\sum_{a \in \text{Por} - \varphi_k} D_a \leq B$ .
- (P3) (Reward)  $\sum_{j=0}^k \sum_{a \in \text{Por}_j - \varphi_j} \sum_{v \in Q_{a, \text{next}(a)}^* - \text{next}(a)} \pi_v(2^j - 1) \geq OPT/8$ .
- (P4) (Prefix-size)  $\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_h} \mu^j(Q_{a, \text{next}(a)}^* - \text{next}(a)) \leq (K + 1)2^j$  for all  $j \in \llbracket k \rrbracket$ .
- (P5) (Size)  $\mu^j(Q_{a,b}^* - b) \leq 2^j$  for every  $j \in \llbracket k \rrbracket$  and pair of consecutive nodes  $a, b \in \text{Por}_j$ .

**Configuration LP and non-adaptive algorithm.** Now assume that we have found, by enumeration, nodes  $\varphi_0, \dots, \varphi_k$ , where  $k \leq L$ , ordered node-sets  $\text{Por}_j$  for  $j \in \llbracket k \rrbracket$ , and length bounds  $D_a$  for every pair of consecutive nodes  $a, b \in \text{Por} := \bigcup_{j=0}^k \text{Por}_j$ , as stipulated by Theorem 4.2. (We also need to enumerate for  $\{m_a, \gamma_a\}_{a \in \text{Por} - \varphi_k}$ ; we do not use these quantities directly, but these are used to specify the  $D_a$  length bounds.) That is, these objects are compatible with suitable  $Q_{a,b}^*$  paths such that P1–P5 hold. Clearly, this enumeration takes  $(n \log B)^{O(N_1 L)} = (n \log B)^{O(\log W \log \log W)}$  time, which is the source of the running time in Theorem 1a.



■ **Figure 2** Portal nodes  $\text{Por}$  and paths between portal nodes. The solid nodes depict  $\text{Por}_j$ .

We formulate a configuration LP to find  $a$ - $b$  paths, for every pair of consecutive nodes  $a, b \in \text{Por}$ , satisfying properties P1, P3–P5. To this end, fix some  $j \in \llbracket k \rrbracket$  and  $a \in \text{Por}_j - \varphi_j$ , and let  $b = \text{next}(a)$ . The valid  $a$ - $b$  paths (i.e., the configurations) are the solutions to the following (deterministic) *point-to-point knapsack orienteering* ( $\text{KnapOrient}$ ) problem: the end-nodes are  $a, b$ , the length budget is  $D_a$ , the knapsack weights are  $\mu_v^j$  for all  $v \in V - b$  and  $\mu_b^j = 0$ , and the knapsack-budget is  $2^j$ . Let  $\mathcal{I}_a$  denote the set of all feasible solutions to this  $\text{KnapOrient}$  instance.

The configuration LP has variables  $x_\tau^a$ , for every  $a \in \text{Por} - \varphi_k$  and  $\tau \in \mathcal{I}_a$ , indicating the  $a$ - $\text{next}(a)$  paths that are chosen.

$$\max \sum_{j=0}^k \sum_{a \in \text{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a} x_\tau^a \cdot \left( \sum_{v \in \tau - \text{next}(a)} \pi_v (2^j - 1) \right) \quad (\text{CKO-P})$$

$$\text{s.t.} \quad \sum_{\tau \in \mathcal{I}_a} x_\tau^a = 1 \quad \forall a \in \text{Por} - \varphi_k \quad (1)$$

$$\sum_{a \in \text{Por} - \varphi_k} \sum_{\tau \in \mathcal{I}_a: v \in \tau - \text{next}(a)} x_\tau^a \leq 1 \quad \forall v \in V \quad (2)$$

$$\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_h} \sum_{\tau \in \mathcal{I}_a} x_\tau^a \cdot \mu^j(\tau - \text{next}(a)) \leq (K+1)2^j \quad \forall j \in \llbracket k \rrbracket \quad (3)$$

$$x \geq 0.$$

Constraints (1) encodes that we select an  $a$ - $b$  path for every consecutive pair of nodes  $a, b \in \text{Por}$ , and constraints (2) ensure that each node  $v$  lies on at most one of these  $a$ - $b$  paths; constraint (3) encodes the (Prefix-size) property P4. (Note that if  $\varphi_{h-1} = \varphi_h$ , then  $\text{Por}_h = \{\varphi_h\}$ , so we do not have any term for index  $h$  in the objective function, and on the LHS of (3).)

To gain some intuition, notice that Theorem 4.2 shows that there is a feasible integral solution to (CKO-P) of objective value at least  $OPT/8$ : we set  $x_\tau^a = 1$  for  $\tau = Q_{a, \text{next}(a)}^*$  for every  $a \in \text{Por} - \varphi_k$ . Properties P1 and P5 show that  $Q_{a, \text{next}(a)}^* \in \mathcal{I}_a$ ; property P4 shows that (3) holds, and P3 shows that the objective value is at least  $OPT/8$ .

We can solve (CKO-P) approximately, given an approximation algorithm for  $\text{KnapOrient}$ , since this can be used to obtain an approximate separation oracle for the dual of (CKO-P).

▷ Claim 4.3. The optimal value of (CKO-P),  $OPT_{\text{CKO-P}}$ , is at least  $OPT/8$ .

► **Lemma 4.4.** *Given an  $\alpha$ -approximation algorithm for KnapOrient, we can compute in polytime a solution  $\bar{x}$  to (CKO-P) of objective value at least  $OPT_{\text{CKO-P}}/\alpha$ .*

We use randomized rounding to round the solution  $\bar{x}$  obtained by Lemma 4.4, and Chernoff bounds yield that this only incurs an  $O(1)$ -factor loss in the objective, and in the violation of constraints (3); here is where we crucially exploit property P5. We then obtain an  $O(K)$ -approximate non-adaptive policy for CorrKO from the rounded solution.

■ **Algorithm CSKO-ALG.** // Rounding  $(\bar{x}, \bar{y})$  and obtaining a non-adaptive policy

- 
- 1 Independently, for each  $a \in \text{Por} - \varphi_k$ , letting  $b = \text{next}(a)$ , do the following: pick an  $a$ - $b$  path by choosing  $\tau \in \mathcal{I}_a$  with probability  $\bar{x}_\tau^a/2$ , and choosing the “direct” path  $a, b$  with the remaining probability 0.5; let  $P_{a,b}$  denote the path picked.
  - 2 If for any  $j \in \llbracket k \rrbracket$ , we have  $\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_h} \mu^j (P_{a, \text{next}(a)} - \text{next}(a)) > 5(K+1)2^j$ , then **return** the empty policy that does not visit any node.
  - 3 Consider the concatenated sequence of nodes  $\{P_{a, \text{next}(a)}\}_{a \in \text{Por} - \varphi_k}$  (where  $\text{Por}$  is ordered as in Theorem 4.2). If a non-portal node is repeated in this sequence, then shortcut the  $P_{a, \text{next}(a)}$  paths so as to retain only the first occurrence of each node. Let  $P'_{a, \text{next}(a)}$  denote the shortcut version of  $P_{a, \text{next}(a)}$  (which is still an  $a$ - $\text{next}(a)$  path). Let  $P'$  be the rooted path given by the node-sequence  $\{P'_{a, \text{next}(a)}\}_{a \in \text{Por} - \varphi_k}$ , where we retain only one copy of each portal node.
  - 4 Sample each  $v \in P' - \rho$  independently with probability  $\frac{1}{10(K+1)}$  to obtain the rooted path,  $P''$ . **return** the non-adaptive policy  $P''$ .
- 

**Analysis overview.** The key observation is that since for any  $j \in \llbracket k \rrbracket$ , any  $h \leq j$ , any  $a \in \text{Por}_h - \varphi_h$ , and any  $\tau \in \mathcal{I}_a$ , we have  $\mu^j(\tau - \text{next}(a)) \leq 2^j$ , we obtain that  $\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_h} \mu^j (P_{a, \text{next}(a)} - \text{next}(a))$  is the sum of a collection of independent  $2^j$ -bounded random variables,<sup>2</sup> whose expectation is  $O((K+1) \cdot 2^j)$ , due to constraint (3). It follows from Chernoff bounds that the probability that this sum exceeds  $5(K+1)2^j$ , for any fixed index  $j$ , is  $\exp -\Omega(K)$ , and so by a union bound, step 2 succeeds with high probability (Lemma 4.5).

To bound the reward obtained, consider a node  $v$  and index  $j \in \llbracket k \rrbracket$ , and define  $\bar{y}_v^j := \sum_{a \in \text{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a: v \in \tau - \text{next}(a)} \bar{x}_\tau^a$ . (Note that  $\sum_{h=0}^k \bar{y}_v^h \leq 1$ .) We say that  $v$  is “visited by segment  $j$ ” if  $v \neq \varphi_j$  and  $v \in \bigcup_{a \in \text{Por}_j - \varphi_j} P_{a, \text{next}(a)}$ ; we say that  $v$  is “retained by segment  $j$ ” if  $v \neq \varphi_j$  and  $v$  remains on  $\bigcup_{a \in \text{Por}_j - \varphi_j} P'_{a, \text{next}(a)}$  after the shortcutting in step 3. Note that the latter events are disjoint, for different  $j$ s. (Note that for a portal node in  $\text{Por}_j - \varphi_j$ , both events happen with probability 1.) Clearly,  $v$  is retained by segment  $j$  only if it is visited by segment  $j$ . For convenience of analysis, we will view step 3 as being executed even if step 2 fails, so we can talk about the event “ $v$  retained by segment  $j$ ” regardless of the outcome of step 2. It is not hard to argue that  $\Pr[v \text{ is retained by segment } j] = \Omega(\bar{y}_v^j)$ , but we need some care to show that this holds even when we condition on the event that step 2 succeeds, as subtle dependencies between events arise here. Nevertheless, we show that this indeed holds (Lemma 4.6).

<sup>2</sup> This is the key difference from [2]. They guess only the  $\varphi_j$  nodes, and so in their case, the corresponding sum gets decomposed into the sum of  $(K+1)2^j$ -bounded random variables, and so an application of Chernoff bounds incurs an additional  $\frac{\log k}{\log \log k} = \frac{\log \log W}{\log \log \log W}$ -factor.

## 29:16 Approximation Algorithms for Correlated Knapsack Orienteering

Finally, given that step 2 succeeds and the rounded path  $P'$  satisfies (3) with  $O((K+1)2^j)$  on the RHS, due to the random sampling in step 4, we can argue that, for any node  $v$  retained by segment  $j$ , the non-adaptive policy processes  $v$  by time  $2^j - 1$  with probability  $\frac{1}{O(K)}$  (Lemma 4.7). Thus, the expected reward of the non-adaptive policy is  $\frac{1}{O(K)} \cdot \sum_{j=0}^k \sum_{v \in V} \bar{y}_j^v \cdot \pi_v(2^j - 1) \geq \frac{OPT}{O(K)}$ .

For an index  $j \in \llbracket k \rrbracket$ , let  $\mathcal{B}_j$  be the event that  $\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_j} \mu^j(P_{a, \text{next}(a)} - \text{next}(a)) > 5(K+1)2^j$ . So  $\mathcal{B} := \bigvee_{j=0}^k \mathcal{B}_j$  is the event that step 2 fails; let  $\mathcal{B}^c$  denote the complement of  $\mathcal{B}$ . Recall that  $K = 3 \log \log(6W) + 12$ .

► **Lemma 4.5.**  $\Pr[\mathcal{B}_j] \leq e^{-(K+1)}$  for all  $j \in \llbracket k \rrbracket$ . Hence,  $\Pr[\mathcal{B}] \leq 1/\text{poly}(\log W)$ .

► **Lemma 4.6.** For any node  $v \in V$  and any  $j \in \llbracket k \rrbracket$ , we have  $\Pr[\{v \text{ is retained by segment } j\} \wedge \mathcal{B}^c] \geq \frac{\bar{y}_j^v}{16}$ .

► **Lemma 4.7.** Consider any node  $v \in V - \varphi_k$ . Suppose that  $v$  is retained by segment  $j$  in step 3. Then  $\Pr[\text{non-adaptive policy } P'' \text{ processes } v \text{ by time } 2^j - 1] \geq \frac{1}{20(K+1)}$ , where the probability is over both the random sampling in step 4 and the random execution of  $P''$ .

**Proof of Theorem 1a.** Combining Lemmas 4.6 and 4.7, and since for any  $v \in V - \varphi_k$ , the events “ $v$  is retained by segment  $j$ ” are disjoint across different  $j$ s, the expected reward obtained from a node  $v$  is at least  $\frac{\sum_{j=0}^k \pi_v(2^j - 1) \bar{y}_j^v}{320(K+1)}$ . So the total expected reward obtained by  $P''$  is at least  $\frac{1}{320(K+1)} \cdot (\text{objective value of } \bar{x}) = OPT/O(K)$ .

The running time is polynomial in the time needed to enumerate the quantities in Theorem 4.2, which is  $\text{poly}((n \log B)^{O(\log W \log \log W)}) = O((n + \log B)^{O(\log W \log \log W)})$ . ◀

### 4.2 Polynomial-time $O(\log W)$ -approximation algorithm

The polytime algorithm also proceeds by gleaning some structural insights from an optimal adaptive policy that enable one to reduce the problem to rooted knapsack orienteering, losing an  $O(\log W)$ -factor. Recall that  $L = \lceil \log W \rceil$ .

► **Theorem 4.8.** There exists an index  $j \in \llbracket L \rrbracket$  such that, for the  $\text{KnapOrient}$ -instance with start node  $\rho$ , travel budget  $B$ , knapsack budget  $2^{j+1}$ , knapsack weights  $\{\mu_v^j\}_{v \in V}$ , and rewards  $\{\pi_v(2^j - 1)\}_{v \in V}$ , the optimal value of the LP-relaxation (KO-LP), is at least  $OPT/(L+1)$ .

**Proof of Theorem 1b.** Theorem 4.8 leads to the following simple algorithm. For the index  $j$  in the theorem statement, we solve (KO-LP) and round it to an *integer solution*  $P$  losing a factor of 5 (see Theorem 2.3). We sample each non-root node in  $P$  independently with probability  $\frac{1}{4}$ , and return the resulting rooted path  $P''$ . To analyze this, for any  $v \in P$ , we have that the probability that the non-adaptive policy  $P''$  processes  $v$  by time  $2^j - 1$  is at least  $\frac{1}{8}$ . The claim follows because  $\Pr[\sum_{w \prec_{P''} v} S_w \geq 2^j] = \Pr[\sum_{w \prec_{P''} v} X_w^j \geq 2^j]$ , which is at most

$$\frac{\mathbb{E}[\sum_{w \prec_{P''} v} X_w^j]}{2^j} = \frac{1}{4} \cdot \frac{\mathbb{E}[\sum_{w \prec_P v} X_w^j]}{2^j} \leq \frac{1}{4} \cdot \frac{\sum_{w \in P} \mu_w^j}{2^j} \leq \frac{1}{2}.$$

The probability of the stated event is therefore at least  $\Pr[v \in P'']/2 \geq 1/8$ . Therefore, the expected reward obtained is at least  $\frac{1}{8} \cdot \sum_{v \in P} \pi_v(2^j - 1) \geq \frac{OPT}{L+1} \cdot \frac{1}{5} \cdot \frac{1}{8} = OPT/O(L)$ . ◀

## 5 Refined approximation guarantees and hardness results for CorrKO

In this section, we perform a fine-grained-complexity study of CorrKO. Motivated by the fact that our adaptivity-gap lower bound for CorrKO utilizes distributions of support-size 3, whereas the adaptivity-gap lower-bound example for stochastic orienteering [2] considers *weighted Bernoulli distributions*, we investigate the complexity of CorrKO when we have distributions supported on at most 2 points – we call this special case 2CorrKO – as also the further special case where the vertex-size distributions are weighted Bernoulli distributions.

In stark contrast with stochastic orienteering, we show that the *adaptivity gap is a constant* for 2CorrKO. Moreover, we obtain non-adaptive  $O(1)$ -approximation algorithms that run in polynomial time for weighted Bernoulli distributions (Theorem 5.5), and in time  $(n + \log B)^{O(\log W)}$  for general 2CorrKO (Theorem 5.4).

The chief insight underlying the above results is that one can isolate a novel *deterministic* VRP, that we call *orienteering with knapsack deadlines* (OrientKD), that governs the complexity of 2CorrKO. In OrientKD, we are given an (rooted or P2P) orienteering instance, along with nonnegative knapsack weights  $\{\text{wt}_v\}_{v \in V}$  and *knapsack deadlines*  $\{\text{KD}_v\}$ . A path  $P$  with start node  $a$  is feasible, if it is feasible for the orienteering instance, and  $\sum_{u \in P_{a,v}} \text{wt}_u \leq \text{KD}_v$  for every node  $v \in P$ ; the goal is to find a feasible path  $P$  that obtains the maximum reward. For this problem, we obtain the following approximation results.

► **Theorem 5.1.** *We can obtain the following approximation guarantees for OrientKD:*

- (a)  $O(1)$ -approximation in  $(n + \log B)^{O(\log W)}$  time;
- (b) polytime  $O(\log(\frac{\max_v \text{KD}_v}{\text{KD}_{\min}}))$ -approximation, where  $\text{KD}_{\min}$  is the minimum non-zero knapsack deadline.

We show that, up to constant factors, OrientKD is *equivalent to 2CorrKO in terms of approximability* (Theorem 5.4). The  $O(1)$ -approximation for 2CorrKO, and the polytime  $O(1)$ -approximation for weighted Bernoulli distributions both fall out as direct consequences of this equivalence: the former, because we can devise an  $(n + \log B)^{O(\log W)}$ -time  $O(1)$ -approximation for OrientKD (Theorem 5.1); the latter, because the OrientKD instance that one needs to solve for weighted Bernoulli distributions is in fact a KnapOrient instance. Another corollary is a hardness result for CorrKO showing that an  $\alpha$ -approximation for CorrKO relative to the *non-adaptive optimum* implies an  $O(\alpha)$ -approximation for OrientKD (Theorem 5.6); this follows because such an approximation guarantee for CorrKO implies an  $O(\alpha)$ -approximation for 2CorrKO (since the adaptivity gap for 2CorrKO is  $O(1)$ ).

**Difficult instances of CorrKO.** We begin by distilling the key source of difficulty for CorrKO (Lemma 5.2). This will prove to be useful when we study 2CorrKO, as it will allow us to focus on the core of the problem. We define the size instantiation  $S_v$  of a vertex  $v$  to be large if  $S_v > W/2$ , and small otherwise. We argue that the difficulty of CorrKO stems from instances where most of the optimal reward comes from *vertices that instantiate to a large size with small probability*.

To make this precise, we introduce some notation. For a vertex  $v$ , we can split its reward  $R_v$  as  $R_v = R_v^{>W/2} + R_v^{\leq W/2}$ , where  $R_v^{>W/2} := R_v \mathbb{1}_{S_v > W/2}$  and  $R_v^{\leq W/2} := R_v \mathbb{1}_{S_v \leq W/2}$ . We can consider the modified CorrKO instances  $\mathcal{I}^{>W/2}$  and  $\mathcal{I}^{\leq W/2}$ , where the rewards are given by  $\{R_v^{>W/2}\}_{v \in V}$  and  $\{R_v^{\leq W/2}\}_{v \in V}$  respectively; so in  $\mathcal{I}^{>W/2}$ , we only collect non-zero reward from large instantiations, and in  $\mathcal{I}^{\leq W/2}$ , we only collect non-zero reward from small instantiations. For  $p \in [0, 1]$ , define  $\mathcal{I}^{>W/2}(p)$  to be the instance with vertex set  $V(p) := \{v \in V : \Pr\{S_v > W/2\} \leq p\}$  (note that  $p \in V(p)$ ). Thus, in instance  $\mathcal{I}^{>W/2}(p)$ , we only consider vertices that instantiate to a large size with probability at most  $p$  (i.e., small probability), and collect reward only from large instantiations.

► **Lemma 5.2.** *Suppose we have an  $\alpha$ -approximation algorithm for CorrKO instances of the form  $\mathcal{I}^{>W/2}(0.5)$ . Then, we can obtain an  $(\alpha + O(1))$ -approximation algorithm for all CorrKO instances.*

**Proof.** A CorrKO instance  $\mathcal{I}$  can be decomposed into three instances,  $\mathcal{I}_1 = \mathcal{I}^{\leq W/2}$ ,  $\mathcal{I}_2 = \mathcal{I}^{>W/2}(0.5)$ , and  $\mathcal{I}_3$  with vertex set  $V_3 := \{v \in V : \Pr[S_v > W/2] > 0.5\}$  and rewards  $\{R_v^{>W/2}\}_{v \in V_3}$ . Any vertex  $v$  yields positive reward in at most one of these 3 instances for any size instantiation, and so  $OPT = OPT(\mathcal{I}) \leq OPT(\mathcal{I}_1) + OPT(\mathcal{I}_2) + OPT(\mathcal{I}_3)$ .

We can obtain non-adaptive policies that yield approximation guarantees of  $\beta_1 = O(1)$ ,  $\beta_3 = O(1)$  for  $\mathcal{I}_1$  and  $\mathcal{I}_3$  respectively. Any adaptive policy for  $\mathcal{I}_3$  can collect positive reward from at most one vertex, which is the first vertex that instantiates to a large size; after this the policy may as well stop. The expected number of nodes visited by an adaptive policy is at most  $\sum_{i \geq 1} 2^{-(i-1)} \leq 4$ , so simply visiting the node in  $V_3$  with largest expected reward, yields an  $O(1)$ -approximation to  $OPT(\mathcal{I}_3)$ . For  $\mathcal{I}_1$ , one can argue that an  $O(1)$ -approximation follows by solving the KnapOrient instance with rewards  $\{\mathbb{E}[R_v^{\leq W/2}]\}_{v \in V}$ , travel budget  $B$ , knapsack weights  $\{\mathbb{E}[\min\{S_v, W\}]\}_{v \in V}$ , and knapsack budget  $2W$ .

Let  $\beta_2 = \alpha$ . Consider now the algorithm that With probability  $\frac{\beta_j}{\beta_1 + \beta_2 + \beta_3}$ , runs the corresponding algorithm for instance  $\mathcal{I}_j$ , for  $j = 1, 2, 3$ . The expected reward obtained via this is at least  $\sum_{j=1}^3 \frac{\beta_j}{\beta_1 + \beta_2 + \beta_3} \cdot \frac{OPT(\mathcal{I}_j)}{\beta_j} \geq \frac{OPT}{\beta_1 + \beta_2 + \beta_3} = \frac{OPT}{\alpha + O(1)}$ . ◀

## 5.1 2CorrKO: CorrKO with distributions of support-size at most 2

Recall that 2CorrKO denotes the special case of CorrKO where, for each vertex  $v$ , the distribution of  $S_v$  is supported on at most 2 values, denoted  $S_v^{(1)}, S_v^{(2)}$  with  $S_v^{(1)} \geq S_v^{(2)}$ . By Lemma 5.2, to obtain an  $O(1)$ -approximation for 2CorrKO, it suffices to consider the instance  $\mathcal{I}_2 = \mathcal{I}^{>W/2}(0.5)$ , and we focus on such instances in the sequel. To keep notation simple, we continue to use  $V$  to denote the vertex set of  $\mathcal{I}_2$ . Then we may assume that the (size, reward) distribution for each  $v \in V$  is  $(S_v^{(1)}, R_v)$  with probability  $p_v$ , and  $(S_v^{(2)}, 0)$  with probability  $1 - p_v$ , where (i)  $S_v^{(1)} > W/2 \geq S_v^{(2)}$  and (ii)  $p_v \leq 0.5$ . Property (i) holds because if  $S_v^{(1)} \leq W/2$ , then  $v$  yields 0 reward for  $\mathcal{I}_2$ , so may be discarded; if  $S_v^{(1)} > W/2$ , then  $\Pr[S_v > W/2] = 1$ , which means that  $v$  would not be considered for  $\mathcal{I}_2$ . Given (i) the reward when the size is  $S_v^{(2)}$  must be 0, and (ii) holds because  $p_v = \Pr[S_v > W/2]$ . We first argue that the adaptivity gap for such instances is 1.

► **Theorem 5.3.** *The adaptivity gap for 2CorrKO (instances of the form  $\mathcal{I}^{>W/2}(0.5)$ ) is 1.*

**Proof.** Let  $\mathcal{T}$  be the decision tree of an optimal adaptive policy. Consider the (rooted) path  $\sigma$  of  $\mathcal{T}$  corresponding to the  $S_v^{(2)}$  size instantiations. Then  $\mathcal{T}$  cannot collect any reward outside of  $\sigma$ , since the residual knapsack budget when we reach any node  $v \in \mathcal{T} \setminus \sigma$  is less than  $W/2$ . So the non-adaptive policy represented by  $\sigma$  has the same expected reward as  $\mathcal{T}$ . ◀

We now show that the resulting 2CorrKO problem is equivalent to OrientKD, up to constant-factor approximation losses. By “equivalent”, we always mean equivalent up a multiplicative  $O(1)$  factor. We actually show that 2CorrKO is equivalent to another problem, *knapsack orienteering with knapsack deadlines* (KnapOrientKD), which is the knapsack-constrained version of OrientKD; by Theorem 2.1, OrientKD and its knapsack-constrained version KnapOrientKD are equivalent, so this implies that 2CorrKO and OrientKD are equivalent.



► **Theorem 5.4.** *Given an  $\alpha$ -approximation algorithm for one of the problems, KnapOrientKD or 2CorrKO, one can obtain an  $O(\alpha)$ -approximation algorithm for the other. Hence, the problems 2CorrKO and OrientKD are equivalent. This implies an  $O(1)$ -approximation algorithm for 2CorrKO with running time  $(n + \log B)^{O(\log W)}$ .*

The approximation guarantee above follows from the guarantee for OrientKD stated in Theorem 5.1. We briefly sketch how to reduce 2CorrKO to KnapOrientKD. By essentially “inverting” this reduction, we obtain the opposite reduction, from KnapOrientKD to 2CorrKO.

Let  $\mathcal{I}$  be a 2CorrKO instance. By Theorem 5.3, we can focus on non-adaptive policies for  $\mathcal{I}$ . Let  $\tau$  be a  $\rho$ -rooted path representing a non-adaptive policy. It is not hard to show that the expected reward from a node  $v \in \tau$  is  $p_v R_v \prod_{w \prec_\tau v} (1 - p_w)$  if  $\sum_{w \prec_\tau v} S_w^{(2)} \leq W - S_v^{(1)}$ , and is 0 otherwise. Also, one can argue that the total expected reward from nodes  $v \in \tau$  with  $\sum_{w \prec_\tau v} p_w > 1$  is a small fraction of  $OPT(\mathcal{I})$ . This motivates the following reduction to KnapOrientKD. We set rewards  $\{\pi_v R_v\}_{v \in V}$ . The constraint  $\sum_{w \prec_\tau v} S_w^{(2)} \leq W - S_v^{(1)}$  can be encoded by a knapsack deadline, by considering knapsack weights  $\{S_w^{(2)}\}_{w \in V}$  and knapsack deadlines  $\{W - S_w^{(1)} + S_w^{(2)}\}_{w \in V}$ . The additional knapsack constraint will encode that the total  $p_w$ -weight of the path should be at most 1, so that the expected reward obtained for  $\mathcal{I}$  from each vertex  $v$  on the KnapOrientKD-solution is  $\Omega(p_v R_v)$ .

For weighted Bernoulli size distributions, which is the special case of 2CorrKO where  $S_v^{(2)} = 0$  for all  $v \in V$ , the above reduction actually crates a KnapOrient instance, since the knapsack-deadlines are trivially satisfied by any rooted path. Since we have a polytime  $O(1)$ -approximation for KnapOrient, we obtain the following.

► **Theorem 5.5** (Weighted Bernoulli size distributions). *There is a polytime  $O(1)$ -approximation for CorrKO with weighted Bernoulli size distributions.*

As noted earlier, combining the equivalence of 2CorrKO and OrientKD, and the  $O(1)$  adaptivity gap for 2CorrKO, yields the following hardness result.

► **Theorem 5.6** (Hardness of approximating the non-adaptive optimum). *Given an  $\alpha$ -approximation algorithm for CorrKO with respect to the non-adaptive optimum, we can obtain an  $O(\alpha)$ -approximation algorithm for OrientKD.*

## 6 CorrKO with cancellations

In CorrKO *with cancellations* (CorrKO-Cancel), the input is the same as in CorrKO, but we are now allowed to *cancel* the processing of the current vertex  $v$  at any (integer) timestep before its size and reward get fully realized; if  $v$  is cancelled, then no reward is collected from  $v$  and we cannot process  $v$  again. As with CorrKO, we only collect reward from vertices that complete by the processing-time horizon  $W$ . Gupta et al. [18] showed that even for correlated knapsack (which is the special case of CorrKO where all vertices are co-located), the optimal reward when we allow cancellations can be substantially larger than the optimal reward without cancellations, so we need to develop new algorithms to handle cancellations.

We obtain the same guarantees for CorrKO-Cancel as for CorrKO: that is,  $O(\log \log W)$ -approximation in  $(n + \log B)^{O(\log W \log \log W)}$  time, and a polytime  $O(\log W)$ -approximation.

We proceed as follows. Recall that for a vertex  $v$ , we define  $R_v^{>W/2} := R_v \mathbb{1}_{S_v > W/2}$  and  $R_v^{\leq W/2} := R_v \mathbb{1}_{S_v \leq W/2}$ . Let  $\mathcal{I}^{>W/2}$  and  $\mathcal{I}^{\leq W/2}$  denote the CorrKO-Cancel instances where the rewards are given by  $\{R_v^{>W/2}\}_{v \in V}$  and  $\{R_v^{\leq W/2}\}_{v \in V}$  respectively. As observed by [18], cancellations do not help for the instance  $\mathcal{I}^{>W/2}$ , i.e., the optimal reward is the same both with and without cancellations. This is because if a policy cancels a vertex  $v$  after it has

run for some  $t \leq W/2$  time steps, we can modify the policy to not process  $v$  at all, without decreasing the reward accrued from subsequently-processed vertices; if  $v$  is cancelled after it has run for more than  $W/2$  time steps, then both with and without cancellation, the policy cannot collect any further reward.

We show that we can obtain an  $O(1)$ -approximation for  $\mathcal{I}^{\leq W/2}$ . With probability 0.5 each, we can work on the instance  $\mathcal{I}^{\leq W/2}$ , where we utilize this  $O(1)$ -approximation, or the instance  $\mathcal{I}^{> W/2}$ , where we utilize the approximation results for CorrKO. So this yields: an  $O(\log \log W)$ -approximation in quasi-polytime, and a polytime  $O(\log W)$ -approximation.

So we focus on obtaining an  $O(1)$ -approximation for CorrKO-Cancel instances of the form  $\mathcal{I}^{\leq W/2}$ . Our approach is based on LP-rounding, by combining the LP-rounding approaches for orienteering in [14] and the correlated knapsack problem with cancellations in [18]. We combine the LP-relaxations for these two problems to obtain the following LP, whose optimal value yields an upper bound on the optimal reward. We use  $R_u(t)$  to denote the reward  $R_u$  when the size  $S_u$  is  $t$ ; this is 0 if  $\Pr[S_u = t] = 0$ . Note that  $R_u(t) = 0$  for all  $t > W/2$ , since we are considering  $\mathcal{I}^{\leq W/2}$ .

$$\begin{aligned}
\max \quad & \sum_{u \in V} \sum_{t=1}^{W/2} z_{u,t} \cdot \Pr[S_u = t \mid S_u \geq t] \cdot R_u(t) && \text{(CKOC-LP)} \\
\text{s.t.} \quad & \text{(O1)–(O5)} \\
& \sum_{v \in V} z_u^v = z_{u,0} && \forall u \in V && (4) \\
& z_{u,t} = s_{u,t} + z_{u,t+1} && \forall u \in V, t \in \llbracket W \rrbracket && \text{(CK1)} \\
& s_{u,t} \geq \Pr[S_u = t \mid S_u \geq t] \cdot z_{u,t} && \forall u \in V, t \in \llbracket W \rrbracket && \text{(CK2)} \\
& \sum_{u \in V} \sum_{t=0}^W t \cdot s_{u,t} \leq W && && \text{(CK3)} \\
& x, z, s \geq 0.
\end{aligned}$$

The  $x_a^v$  and  $z_u^v$  variables, and constraints (O1)–(O5) are from the LP for rooted orienteering (and also present in LP (KO-LP) for KnapOrient). They encode the arcs included, and vertices visited, respectively by the rooted path, provided that  $v$  is the furthest node from  $\rho$  that is visited. Constraints (O1)–(O5) are valid because any rooted path  $Q$ , corresponding to an execution of an adaptive policy, satisfies these constraints, where the superscript  $v$  in the non-zero variables is the furthest node from  $\rho$  on  $Q$ .

The  $z_{u,t}$  and  $s_{u,t}$  variables, constraints (CK1)–(CK3), and the objective function are from the LP in [18] for correlated knapsack with cancellations. For any vertex  $u$  and  $t \geq 0$ , variable  $z_t^u$  encodes that  $u$  is processed for a least  $t$  time units, and  $s_{u,t}$  encodes that  $u$  is processed for *exactly*  $t$  time units. Thus, variable  $z_{u,0}$  encodes that  $u$  is visited, and constraint (4) links the orienteering and correlated knapsack LPs. Gupta et al. [18] show (see Theorem 3.1 in [18]) that constraints (CK1)–(CK3) are valid for correlated knapsack with cancellations, and that the objective function provides an upper bound on the expected reward obtained.

We remark that [18] showed that one can replace (CK1)–(CK3) with a polynomial-size formulation losing an  $O(1)$ -factor, which applies here as well.

We round an optimal solution  $(\bar{x}, \bar{z}, \bar{s})$  to (CKOC-LP) in two phases. We first *extract a suitable knapsack orienteering instance from the LP solution*, and use Theorem 2.3 to obtain a good rooted path  $Q$  for this KnapOrient instance. Now, we select a subsequence of  $Q$  to visit by solving a correlated knapsack with cancellations problem involving only vertices in  $Q$ . The KnapOrient-instance is set up so that from  $(\bar{x}, \bar{z}, \bar{s})$ , one can extract a good LP solution to the

correlated knapsack problem restricted to vertices in  $Q$ . We utilize the LP-rounding result in [18] to round this solution to obtain a CorrKO-Cancel solution that visits the vertices in  $Q$  in order, potentially cancelling some vertices along the way. Thus, we obtain a non-adaptive policy for CorrKO-Cancel. In the second phase, we crucially leverage an important aspect of the LP-rounding algorithm in [18] for correlated knapsack with cancellations, namely that it is *order oblivious*: its guarantee does not depend on the order in which the vertices (i.e., items in correlated knapsack) are considered. This flexibility allows us to consider vertices in  $Q$  in the order they are visited, and thereby ensure that the travel-budget constraint is satisfied. (We remark that for the correlated knapsack without cancellations, we do not have this flexibility, when considering large instantiations; see Appendix A. This lack of flexibility is the main obstacle in obtaining a good solution from large instantiations in CorrKO.)

## 7 $O(\log \log B)$ -approximation for CorrO

Our approach in Section 4 for CorrKO can be utilized to yield the guarantees mentioned in Theorem 1.3: that is, an  $O(\alpha \log \log B)$ -approximation algorithm for CorrO in time  $(n + \log B)^{O(\log B \log \log B)} \cdot T$ , where  $T$  is the running time of the given  $\alpha$ -approximation algorithm for deadline TSP. The algorithm in [15] for deadline TSP translates to an  $O(1)$ -approximation in  $n^{O(\log B)}$  time, so this implies an  $O(\log \log B)$ -approximation for CorrO in quasi-polytime. We also simplify the exposition significantly by making use of monotone-reward TSP [15] as a subroutine.

Let  $K = 3 \log(6 \log B) + 12$ ,  $L = \lceil \log B \rceil$ ,  $N_1 = 2(K + 1)$ . Define  $\varphi_{-1} := \rho$ , and  $\pi_v(t) := \mathbb{E}[R_v \cdot \mathbb{1}_{S_v \leq B-t}] = \sum_{t'=0}^{B-t} \Pr[S_v = t'] \cdot \mathbb{E}[R_v | S_v = t']$ . Let  $OPT^{\text{CorrO}}$  denote the optimal reward for the CorrO instance. We may assume that  $\pi_v(d_{\rho,v}) \leq OPT^{\text{CorrO}}/4$  for every  $v \in V$ , as otherwise, we can obtain  $\Omega(OPT^{\text{CorrO}})$  reward by going to a single node. The algorithm in [2] is based on the following structural result, which we have paraphrased (and corrected slightly) to conform to our notation.

► **Lemma 7.1** (Lemma 3.6 in [2]). *There exists a rooted path  $P$  with  $d(P) \leq B$ , and vertices  $\varphi_0 \preceq \varphi_1 \preceq \dots \preceq \varphi_k$  on  $P$  for some  $k \leq L$ , such that:*

- (a)  $\sum_{j=0}^k \sum_{v \in P_{\varphi_{j-1}, \varphi_j - \varphi_j}} \pi_v(d(P_{\rho,v} + 2^j - 1)) \geq OPT^{\text{CorrO}}/4$ ; and
- (b)  $\mu^j(P_{\rho, \varphi_j} - \varphi_j) \leq (K + 1)2^j$  for all  $j \in \llbracket k \rrbracket$ .

We refine this by subdividing each  $P_{\varphi_{j-1}, \varphi_j}$  subpath into at most  $2(K + 1)$  segments each of  $\mu^j$ -weight at most  $2^j$ , and by guessing the two-point regrets of these segments, to obtain a structural result analogous to Theorem 4.2.

► **Theorem 7.2** (Structural result for CorrO). *Let the rooted path  $P$  and node-sequence  $\varphi_0, \dots, \varphi_k$ , where  $k \leq L$ , be as in Lemma 7.1. For each  $j \in \llbracket k \rrbracket$ , there is a vertex-set  $\text{Por}_j \subseteq P_{\varphi_{j-1}, \varphi_j}$  containing  $\varphi_{j-1}, \varphi_j$ , with  $|\text{Por}_j| \leq N_1$ , whose nodes are ordered by the order they appear on  $P$ , and for every node  $a \in (\bigcup_{j=0}^k \text{Por}_j) - \varphi_k$ , there is a path  $\overline{Q}^a$ , node  $m_a$ , integer  $\gamma_a \geq 0$ , satisfying the following properties. For  $a, b \in \text{Por} := \bigcup_{j=0}^k \text{Por}_j$ , let  $\text{next}(a)$  be the next node in  $\text{Por}$  after  $a$ , for  $a \neq \varphi_k$ ; let  $b \prec a$  if  $b$  comes before  $a$  in  $\text{Por}$ .*

- (C1) (Distance)  $d(\overline{Q}^a) \leq D_a := 2^{\gamma_a} - 1 + d(a, m_a) + d(m_a, \text{next}(a))$  for every  $a \in \text{Por} - \varphi_k$ .
- (C2) (Total-length)  $\sum_{a \in \text{Por} - \varphi_k} D_a \leq B$ .
- (C3) (Reward)  $\sum_{j=0}^K \sum_{a \in \text{Por}_j - \varphi_j} \sum_{v \in \overline{Q}^a - \text{next}(a)} \pi_v(\sum_{b \prec a} D_b + d(\overline{Q}_{a,v}^a) + 2^j - 1) \geq OPT^{\text{CorrO}}/8$ .
- (C4) (Prefix-size)  $\sum_{h=0}^j \sum_{a \in \text{Por}_h - \varphi_h} \mu^j(\overline{Q}^a - \text{next}(a)) \leq (K + 1)2^j$  for all  $j \in \llbracket k \rrbracket$ .
- (C5) (Size)  $\mu^j(\overline{Q}^a - \text{next}(a)) \leq 2^j$  for every  $j \in \llbracket k \rrbracket$  and every  $a \in \text{Por}_j - \varphi_j$ .

We now exploit this structural result in much the same way as in Section 4.1 by setting up a configuration LP to find the  $\overline{Q}^a$ -paths. Note that only the (Reward) property C3 is different from the (Reward) property in Theorem 4.2 for CorrKO, and correspondingly, we now exploit monotone-reward TSP to capture the reward of an  $\overline{Q}^a$  path. Assume that we have found the “portal nodes”  $\text{Por}$  and length bounds  $D_a$  for all  $a \in \text{Por} - \varphi_k$  satisfying Theorem 7.2. To capture the reward obtained from an  $a$ -next( $a$ ) path, the configurations  $\mathcal{I}_a$  for a node  $a \in \text{Por} - \varphi_k$  will now consist of feasible solutions to P2P *knapsack monotone-reward TSP*, which is the knapsack-constrained version of P2P-monotone-reward TSP: they are simply all  $a$ -next( $a$ ) paths  $\tau$  with  $\mu^j(\tau - \text{next}(a)) \leq 2^j$ . The length budget  $D_a$  will be captured implicitly, by defining the reward function of a node  $v \neq \text{next}(a)$  to be  $\pi_v(\sum_{b \prec a} D_b + d(\tau_{a,v}) + 2^j - 1)$  if  $d(\tau_{a,v}) \leq D_a - d(v, \text{next}(a))$  and 0 otherwise, which is a non-increasing function of  $d(\tau_{a,v})$ ; for  $v = \text{next}(a)$ , the reward function is defined to be identically 0. For notational convenience, define  $\pi^{a,j}(\tau)$  to be the total reward obtained from nodes in  $\tau$  under the above rewards.

The configuration LP for CorrO now has the same constraints as (CKO-P), but the objective function changes to  $\max \sum_{j=0}^k \sum_{a \in \text{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a} x_\tau^a \cdot \pi^{a,j}(\tau)$ . Let (CO-P) denote this LP, and  $OPT_{\text{CO-P}}$  denote its optimal value.

We now have  $OPT_{\text{CO-P}} \geq OPT^{\text{CorrO}}/8$ , and one can argue that an  $\alpha$ -approximation algorithm for deadline TSP (and hence, monotone-reward TSP [15]) can be used to obtain a (CO-P)-solution  $\bar{x}$  of value at least  $OPT_{\text{CO-P}}/O(\alpha)$ . The LP-rounding algorithm and conversion to a non-adaptive policy are *exactly* as in Algorithm CSKO-Alg. The analysis is similar, but we now analyze the reward on a path-by-path basis, considering the reward obtained from paths  $\tau \in \mathcal{I}_a$ , for each  $a \in \text{Por} - \varphi_k$ . We can again argue that step 2 succeeds with high probability, and moreover that the expected reward obtained is large conditioned on this. Hence, we obtain an  $O(K)$  approximation.

---

## References

- 1 Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 166–174, 2004. doi:10.1145/1007352.1007385.
- 2 Nikhil Bansal and Viswanath Nagarajan. On the adaptivity gap of stochastic orienteering. *Mathematical Programming*, 154(1-2):145–172, December 2015. doi:10.1007/s10107-015-0927-9.
- 3 Anand Bhalgat. A  $(2 + \epsilon)$ -approximation algorithm for the stochastic knapsack problem. *Unpublished manuscript*, 2011.
- 4 Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171, 1994. doi:10.1145/195058.195125.
- 5 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM Journal on Computing*, 37(2):653–670, January 2007. doi:10.1137/050645464.
- 6 Deeparnab Chakrabarty and Chaitanya Swamy. Facility Location with Client Latencies: Linear Programming Based Techniques for Minimum Latency Problems. *Mathematics of Operations Research*, 41(3):865–883, 2016. doi:10.1287/moor.2015.0758.
- 7 Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora’s box with correlations: Learning and approximation. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1214–1225, 2020.
- 8 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms*, 8(3):1–27, July 2012. doi:10.1145/2229163.2229167.

- 9 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33(4):945–964, November 2008. doi:10.1287/moor.1080.0330.
- 10 Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Submodular Set Cover with Applications to Boolean Function Evaluation and Min-Knapsack. *ACM Transactions on Algorithms*, 12(3):1–28, June 2016. doi:10.1145/2876506.
- 11 Alina Ene, Viswanath Nagarajan, and Rishi Saket. Approximation Algorithms for Stochastic k-TSP. In *Proceedings of the 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 27:27–27:14, 2017. arXiv:1610.01058.
- 12 Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The  $k$ -traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4):40, November 2007. doi:10.1145/1290672.1290677.
- 13 Zachary Friggstad and Chaitanya Swamy. Approximation algorithms for regret-bounded vehicle routing and applications to distance-constrained vehicle routing. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 744–753, 2014. doi:10.1145/2591796.2591840.
- 14 Zachary Friggstad and Chaitanya Swamy. Compact, Provably-Good LPs for Orienteering and Regret-Bounded Vehicle Routing. In *Proceedings of 19th IPCO*, pages 199–211, 2017.
- 15 Zachary Friggstad and Chaitanya Swamy. Constant-Factor Approximation to Deadline TSP and Related Problems in (Almost) Quasi-Polytime. In *Proceedings of 48th ICALP*, pages 67:1–67:21, 2021.
- 16 Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987. doi:10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.
- 17 Sudipto Guha and Kamesh Munagala. Multi-armed Bandits with Metric Switching Costs. In *Proceedings of 36th ICALP*, pages 496–507, 2009. doi:10.1007/978-3-642-02930-1\_41.
- 18 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and Ramamoorthi Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *52nd Annual Symposium on Foundations of Computer Science*, pages 827–836, 2011.
- 19 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running Errands in Time: Approximation Algorithms for Stochastic Orienteering. *Mathematics of Operations Research*, 40(1):56–79, 2015. doi:10.1287/moor.2014.0656.
- 20 Haotian Jiang, Jian Li, Daogao Liu, and Sahil Singla. Algorithms and Adaptivity Gaps for Stochastic k-TSP. In *Proceedings of 11th ITCs*, pages 45:1–45:25, 2020. arXiv:1911.02506.
- 21 Will Ma. Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1154–1163, 2014. doi:10.1137/1.9781611973402.85.
- 22 Viswanath Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, March 2012. doi:10.1002/net.20435.

## **A** Adversarial orderings can be arbitrarily bad for correlated knapsack

Consider an instance of correlated stochastic knapsack on the set of items  $[n]$  with budget  $W > 2^{n+1}$ . Let  $S_i$  and  $R_i$  denote respectively the random size and random reward of  $i$ , which follows the following distribution.

$$(S_i, R_i) = \begin{cases} (S_i^{(1)} := W - 2^{n-i} + 1, R_i^{(1)} := 1) & \text{with probability } \frac{1}{n} \\ (S_i^{(2)} := 2^{n-i}, R_i^{(2)} := 0) & \text{with probability } 1 - \frac{1}{n}. \end{cases}$$

At most one item can obtain positive reward since  $W/2 < W - 2^{n-i+1} + 1$  for all  $i \in [n]$ .




## 29:24 Approximation Algorithms for Correlated Knapsack Orienteering

Suppose that we are forced to process the items in the ordering  $1, \dots, n$  deciding at each step whether we attempt to insert the current item into the knapsack or abandon it forever. Let  $j$  be the first item that we choose to insert into the knapsack. It instantiates to size  $2^{n-j}$  with probability  $1 - 1/n$ . If this happens we get zero total reward: the residual budget becomes  $W - 2^{n-j}$ , which is less than the  $S_i^{(1)}$ -sizes of items  $j + 1, \dots, n$  (which yield positive reward). Therefore, by processing the items in this ordering the expected reward is at most  $1/n$ . But suppose we process the items in the reverse order  $n, \dots, 1$ . If we attempt to insert items  $n, n - 1, \dots, j$  and get zero reward from all of them, the residual budget is  $W - \sum_{k=j}^n 2^{n-k} = W - 2^{n-j+1} + 1 > S_{j-1}^{(1)}$ , which means that item  $j - 1$  can be inserted and would yield reward 1 with probability  $\frac{1}{n}$ . Thus, the probability that no item gives positive reward is  $(1 - 1/n)^n \leq e^{-1}$  and the expected reward we obtain in this case is at least  $(1 - e^{-1})$ . This is  $\Omega(n)$  times larger than the expected reward that can be obtained by any policy that is forced to process items in the order  $1, \dots, n$ .

# Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem

Gabriel Arpino   

University of Cambridge, UK

Daniil Dmitriev   

ETH Zürich and ETH AI Center, Switzerland

Nicolo Grometto 

Princeton University, USA

---

## Abstract

Consider the **Hitting Set** problem where, for a given universe  $\mathcal{X} = \{1, \dots, n\}$  and a collection of subsets  $\mathcal{S}_1, \dots, \mathcal{S}_m$ , one seeks to identify the smallest subset of  $\mathcal{X}$  which has a nonempty intersection with every element in the collection. We study a probabilistic formulation of this problem, where the underlying subsets are formed by including each element of the universe independently with probability  $p$ . We rigorously analyze integrality gaps between linear programming and integer programming solutions to the problem. In particular, we prove the absence of an integrality gap in the sparse regime  $mp \lesssim \log n$  and the presence of a non-vanishing integrality gap in the dense regime  $mp \gg \log n$ . Moreover, for large enough values of  $n$ , we look at the performance of Lovász's celebrated **Greedy** algorithm [12] with respect to the chosen input distribution, and prove that it finds optimal solutions up to multiplicative constants. This highlights separation of **Greedy** performance between average-case and worst-case settings.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Combinatorial optimization; Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures

**Keywords and phrases** Hitting Set, Random Hypergraph, Integrality Gap, Greedy Algorithm

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.30

**Category** APPROX

**Related Version** *Extended Version*: <https://arxiv.org/abs/2305.05565> [1]

**Acknowledgements** The authors thank Dylan J. Altschuler, Afonso S. Bandeira, Raphaël Barboni, and Anastasia Kireeva for helpful discussions. DD is supported by ETH AI Center doctoral fellowship and ETH Foundations of Data Science initiative. GA is supported by the Cambridge Trust. NG is grateful for the funding received from Elizaveta Rebrova.

## 1 Introduction

**Hitting Set** is a classical problem in combinatorial optimization which, for a given ground set  $\mathcal{X} := \{1, \dots, n\}$  of elements and a collection  $\mathcal{C} := \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$  of subsets of  $\mathcal{X}$ , asks to identify the smallest set  $\mathcal{S} \subseteq \mathcal{X}$  that intersects every subset in  $\mathcal{C}$ . **Hitting Set** arises naturally from the study of *Minimum Vertex Covers on Hypergraphs* (MVCH), upon viewing hyperedges as subsets and vertices as elements of the ground set. This is also known as the *Set Cover* problem [14], which has a rich history in worst-case computational complexity theory, including appearing as one of Karp's 21 NP-complete problems. An important question regards the behaviour of natural random instances of **Hitting Set** where each element of the ground set is independently assigned to any subset with probability  $p$ , motivated, among others, by applications such as group testing [10]. A classical theorem of Lovász [12] gives



© Gabriel Arpino, Daniil Dmitriev, and Nicolo Grometto;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 30; pp. 30:1–30:22



Leibniz International Proceedings in Informatics

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

an upper bound on the integrality gap in this problem which grows with the degree of the underlying hypergraph, i.e., the maximum number of subsets intersecting any one element. This bound was shown to be tight in the worst-case, but leaves much to be desired from an average-case perspective.

In this paper, we characterize the average-case integrality gap present in random **Hitting Set** and prove that, with high probability, Lovász's greedy algorithm [12] finds the minimal hitting set in polynomial time. Namely, we consider the following integer programming (IP) formulation of the problem,

$$\text{val}_{\text{IP}} := \begin{cases} \text{minimize} & \|\mathbf{x}\|_1 \\ \text{subject to} & \mathbf{Ax} \geq \mathbf{1}, \mathbf{x} \in \{0, 1\}^n, \end{cases} \quad (1.1)$$

where the  $i$ -th row of  $\mathbf{A} \in \{0, 1\}^{m \times n}$  provides a binary encoding of the membership of the elements of  $\mathcal{X}$  in the set  $\mathcal{S}_i$  and  $\mathbf{1} := (1, \dots, 1) \in \mathbb{R}^m$ . With the vertex cover formulation of the problem at hand, we note that  $\mathbf{A}$  consists of the incidence matrix of the underlying hypergraph. In particular, the constraint  $\mathbf{Ax} \geq \mathbf{1}$  ensures that each set in  $\mathcal{C}$  is hit by a prescribed candidate solution vector. A natural convex relaxation is obtained by allowing fractional solutions, and may be expressed as the following linear program (LP),

$$\text{val}_{\text{LP}} := \begin{cases} \text{minimize} & \|\mathbf{x}\|_1 \\ \text{subject to} & \mathbf{Ax} \geq \mathbf{1}, \mathbf{x} \in [0, 1]^n. \end{cases} \quad (1.2)$$

Whilst clearly  $\text{val}_{\text{LP}} \leq \text{val}_{\text{IP}}$ , tightness need not hold in general. In fact, for  $m = n$  and  $\mathbf{A} \in \{0, 1\}^{n \times n}$  chosen such that each row and column contains exactly  $k$  ones, for some fixed  $1 < k < n$ , an optimal solution is provided by  $\mathbf{x}_{\text{LP}}^* = (1/k, \dots, 1/k)$ , which is not integral, thus leading to a strictly smaller objective whenever  $n/k$  is not an integer. This evidences the existence of a multiplicative *integrality gap*, as we define next.

► **Definition 1.** *Given solutions  $\text{val}_{\text{IP}}$  and  $\text{val}_{\text{LP}}$  to Equation (1.1) and Equation (1.2) respectively, we define multiplicative integrality gap as follows:*

$$\text{IPGAP} := \frac{\text{val}_{\text{IP}}}{\text{val}_{\text{LP}}}. \quad (1.3)$$

In [12], Lovász proved an essentially optimal worst-case upper bound on the **Hitting Set** multiplicative integrality gap:  $\text{IPGAP} \leq 1 + \log d_{\max}$ , where  $d_{\max}$  corresponds to the maximum degree in the underlying hypergraph. This is obtained by analysing the **Greedy** algorithm (Algorithm 1), which constructs a vertex cover by sequentially adding vertices with the highest degree amongst the uncovered edges, and will be discussed in more detail in the next sections. However, in many natural examples, the maximum degree  $d_{\max}$  grows with the number of vertices in the hypergraph, thus leading to progressively worse bounds for increasingly large hypergraphs. Besides being arguably the most natural candidate for solving **Hitting Set**, the greedy algorithm has been shown to be the best possible polynomial time approximation algorithm [15] for the worst-case instances of this classical problem.

Despite extensive work conducted on **Hitting Set** in the last decades, a gap remains in our understanding of the typical performance of linear programming and the greedy algorithm on random problem instances. We hence pose the following questions:

1. Are there integrality gaps in random instances of **Hitting Set**?
2. Can near-optimal solutions be found efficiently?



In the present work, we provide answers to the above questions *with high probability* (w.h.p.) in a non-asymptotic sense, in the setting where the cardinality  $n$  of the ground set  $\mathcal{X}$  is large but finite. We will prove the absence of integrality gaps up to constants in a wide regime of  $n, m, p$ , by conducting an average case analysis of an algorithm that outputs integral covers of matching size to the fractional ones. In addition, a rigorous analysis of the greedy routine will follow by a straightforward reduction. The forthcoming results are valid under the conditions listed below, which will be assumed to hold throughout.

► **Assumption 2.** *We assume that*

1. *Each element  $j \in \mathcal{X}$  is assigned to any subset  $\mathcal{S}_i$ ,  $i \in [m]$  with probability  $p \equiv p(n)$ , independently. That is,  $\mathbf{A} \in \{0, 1\}^{m \times n}$  is such that  $A_{ij} \stackrel{iid}{\sim} \text{Bernoulli}(p)$ ;*
2.  *$n$  is intended to be large but finite;*
3.  *$m \equiv m(n) = \text{poly}(n)$ , i.e.  $\exists c, C > 0$ , such that  $cn^c \leq m \leq Cn^C$  for  $n$  large enough;*
4. *There exist  $\delta \in (0, 1)$ , such that  $p \equiv p(n)$  satisfies  $1/n^\delta \leq p \leq 1/2$ , for all  $n$  large enough.*

Note that in Assumption 2.3, the upper bound is chosen to avoid trivial solutions w.h.p. which arise, for example, in the setting where the number of sets grows exponentially in the cardinality of  $\mathcal{X}$ . In addition, Assumption 2.4 is by no means restrictive, since one can show that for  $m = \text{poly}(n)$  and  $np \ll \log n$ , we have that  $\mathbf{A}$  contains an all-zero row w.h.p., yielding an infeasible solution for IP. The requirement  $p \leq 1/2$  is chosen for technical convenience and can be relaxed to any constant  $p$ , encompassing the regime in [10].

Our contributions stem from the study of the size of the inclusion sets  $I_j := \{i \in [m] : j \in \mathcal{S}_i\}$ , for  $j \in [n]$ , which in the MVCH formulation of the problem at hand correspond to the set of hyperedges incident to any given vertex. The key quantity under study is the average inclusion set size, that is  $\mathbb{E}|I_j| = mp$ , for all  $j$ , under the present distributional assumptions. This quantity exhibits two separate regimes of interest, referred to as the *sparse*,  $mp \ll \log n$ , and *dense*,  $mp \gg \log n$ , regimes. These, in turn, determine the size of the maximum inclusion set, or maximum degree,  $d_{\max} := \max_{j \in [n]} |I_j|$ . We characterize the integrality gap behaviour up to multiplicative constants and analyse Lovász's Greedy algorithm [12] in these two regimes w.h.p as  $n \rightarrow \infty$ . We do this by proving the success of a simple greedy heuristic, the BlockGreedy algorithm (Algorithm 2). Throughout, we use the notation  $\text{val}_{\text{Gr}}$ ,  $\text{val}_{\text{BGr}}$  to denote the size of the hitting set returned by Greedy and BlockGreedy respectively. Below we provide an informal description of the main results which hold with high probability, where  $A(n) \sim B(n)$  denotes that  $cA(n) \leq B(n) \leq CA(n)$  for large enough  $n$  and for some constants  $c, C > 0$ :

### Sparse Regime ( $mp \ll \log n$ )

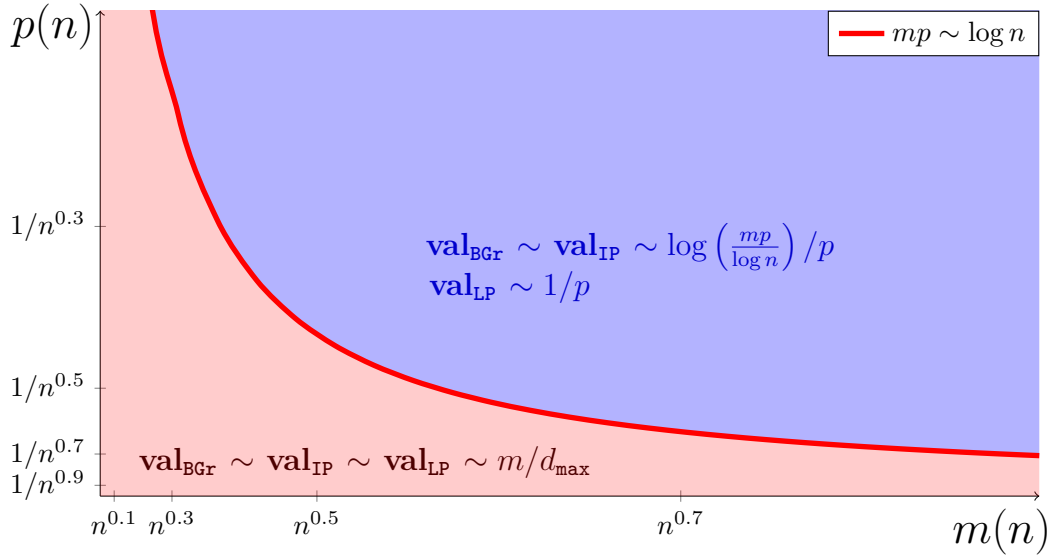
We show that  $\text{IPGAP} \sim 1$  in the sparse regime by proving that the BlockGreedy algorithm succeeds in reaching the LP lower bound of  $\frac{m}{d_{\max}}$ .

$$\text{val}_{\text{BGr}} \sim \text{val}_{\text{IP}} \sim \text{val}_{\text{LP}} \sim \frac{m}{d_{\max}}.$$

### Dense Regime ( $mp \gg \log n$ )

We prove that  $\text{IPGAP} \sim \log \frac{mp}{\log n}$  in the dense regime. We show that the BlockGreedy algorithm performs as well as IP in this regime, i.e.

$$\frac{1}{p} \log \left( \frac{mp}{\log n} \right) \sim \text{val}_{\text{BGr}} \sim \text{val}_{\text{IP}} \gg \text{val}_{\text{LP}} \sim \frac{1}{p} \sim \frac{m}{d_{\max}}.$$



■ **Figure 1** Transition between the sparse and the dense regime for different values of the average inclusion set size  $mp$ .

### Threshold Regime ( $mp \sim \log n$ )

This regime smoothly interpolates between the sparse and dense ones, with  $\text{IPGAP} \sim 1$ . The scaling for all quantities of interest is  $m/d_{\max} \sim 1/p$ .

### Greedy

We prove that  $\text{val}_{\text{Gr}} \sim \text{val}_{\text{IP}}$  when  $\delta < 1/2$ , where  $\delta$  is the parameter from Assumption 2.4.

The results above are also depicted in Figure 1, and the formal statements are given in Corollary 9 and Theorem 10. The rest of the paper is organized as follows. In Section 2, we present relevant notation. In Section 3, we outline and discuss related literature. In Section 4, we prove a number of preliminary results that will be instrumental in developing the core arguments. Subsequently, in Section 5, we delve into the algorithmic aspects of the problem at hand by first providing guarantees for a simple algorithm, **BlockGreedy**. We then analyse **Greedy** by means of a reduction. We conclude in Section 6 by summarizing the results and offering indications for future work. We defer the proofs of more technical results to the appendix, in order to streamline the presentation for the reader's convenience.

## 2 Notation and conventions

For integers  $k \in \mathbb{N}$ , we write  $[k] := \{1, \dots, k\}$ . We denote vectors, matrices by bold-faced Roman letters  $\mathbf{x}, \mathbf{A} \in \mathbb{R}^k, \mathbb{R}^{k \times k}$ , respectively, for some  $k \in \mathbb{N}$ . Define the *inclusion set* of an element, or node,  $j \in [n]$  as  $I_j = \{i \in [m] : j \in \mathcal{S}_i\}$ . We denote the  $\ell_1$  norm of the  $j$ -th column of  $\mathbf{A}$  by  $X_j$ ,  $j \in [n]$ , noting that  $X_j = |I_j|$  and  $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \text{Binomial}(m, p)$ . In addition, we let  $d_{\max} \equiv d_{\max}(X_1, \dots, X_n) := \max_{i \in [n]} X_i$ . We use  $\mathbb{E}, \text{Var}$  to denote expectation and variance, respectively. By  $\lesssim, \gtrsim$  we denote inequalities up to multiplicative constants. We let  $A \sim B$  denote that  $A \lesssim B \lesssim A$  for large enough  $n$ . We let  $\log$  denote the natural logarithm. For possibly random functions  $f(n), g(n)$ , we let  $\{f \lesssim g\}$  denote a sequence of

events  $\{f(n) \leq Ag(n)\}$  for some constant  $A > 0$  independent of  $n$ . Consequently,  $\mathbb{P}(f \lesssim g)$  is viewed as a function of  $n$ . For deterministic functions  $h(n), w(n)$ , we let  $h \ll w, h \gg w$  denote that  $h/w \rightarrow 0, w/h \rightarrow 0$  respectively, as  $n \rightarrow \infty$ . The notation for other inequalities is defined analogously. We say that a sequence of events  $\{A_n\}$  holds *with high probability* (w.h.p.) with respect to a probability measure  $\mathbb{P}$  if there exists a constant  $c > 0$ , independent of  $n$ , such that  $\mathbb{P}(A_n) \geq 1 - n^{-c}$ , for large enough values of  $n$ .

### 3 Related Work

#### Worst-case analysis of Greedy

Perhaps the most well-known algorithm for solving **Hitting Set**, or equivalently **MVCH**, is the greedy algorithm of Lovász [12], with runtime complexity  $O(mn^2)$ . This algorithm, which constructs a cover by sequentially adding elements of the ground set which hit the largest number of remaining subsets, was initially studied by Lovász [12] and Johnson [11] independently, for deterministic hypergraphs. Lovász analyses the greedy algorithm to obtain an upper bound on the **Hitting Set** integrality gap of  $1 + \log d_{\max}$ . Slavik [15] developed the tightest known approximation lower bound for **Greedy**, constructing an instance where **Greedy** finds coverings at least  $\log m$  times as large as the minimum one. Importantly, Feige [6] proved that an approximation ratio of  $(1 - \epsilon) \log m$  is not achievable in polynomial time for any  $\epsilon > 0$  unless  $NP \subset TIME[n^{O(\log \log n)}]$ , certifying **Greedy** as the best possible polynomial-time approximation algorithm for set cover in the worst-case.

#### Random Hitting Set

Little is known about the typical performance of polynomial-time algorithms on random instances of **Hitting Set**. Closing this gap is important from a theoretical standpoint and for applications in combinatorial inference. A prime example of this is found in *group testing*, a classical inference problem where one aims to identify a small subset of defective items within a large population by conducting the smallest number of pooled tests, with applications ranging from the analysis of communication protocols [8] to DNA sequencing [5] and search problems [4]. In [10], Iliopoulos and Zadik consider the smallest hitting set as an estimator in the setting of the group testing problem, referring to it as the *Smallest Satisfying Set* estimator. In particular, they provide extensive empirical evidence supporting the claim that the class of instances of the random hitting set problem induced by non-adaptive group testing is tractably solvable by computers.

#### Insights from Statistical Physics

The analysis of a random instance of **Hitting Set** appears in the work of Mézard and Tarzia and relies on nonrigorous techniques from statistical physics [13]. This work considers regular uniform hypergraphs, where the degree of vertices and the size of edges are fixed and assumed to be constant. Depending on these values, they evidence sharp transitions between three different phases, the so-called replica symmetry, 1-replica symmetry breaking, and full replica symmetry breaking phases, which characterize the complexity of the optimization landscape for this problem in the average case setting.

### Fixed $p$ regime

Another instance was studied by Telelis and Zissimopoulos [16] in the setting of random Bernoulli hypergraphs, where elements belong to subsets independently with *fixed* probability  $p \in (0, 1)$ . Their analysis concerns the asymptotic regime where the size  $n$  of the ground set scales to infinity. In this setting, they study the average-case performance of a simple deterministic algorithm which approximates random **Hitting Set** within an *additive* error term at most  $o(\log m)$  almost everywhere. This gives an improvement over Lovász's argument in [12] which provides a multiplicative bound. However, the analysis in [16] does not capture the case of sparse hypergraphs, i.e., when  $p \rightarrow 0$  as  $n \rightarrow \infty$ . The analysis in [16] also does not prove guarantees for the **Greedy** algorithm in the chosen parameter regime.

### Related problem formulations

We bring to the reader's attention a more recent line of work [2, 3], where the authors obtain bounds on (additive) integrality gaps between the value of a random integer program  $\max \mathbf{c}^T \mathbf{x}, \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \{0, 1\}^n$  with  $m$  constraints and that of its linear programming relaxation for a wide range of distributions on  $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ , holding w.h.p. as  $n \rightarrow \infty$ . These include the case where the entries of  $\mathbf{A}$  are uniformly distributed on an integer interval consisting of at least three elements and where the columns of  $\mathbf{A}$  are distributed according to an isotropic logconcave distribution. However, these fail to capture the setting where  $\mathbf{A}$  is sparse with entries in  $\{0, 1\}$ , which is of interest for **Hitting Set**.

## 4 Preliminary Bounds

In this section, we outline preliminary bounds on  $\text{val}_{\text{LP}}, \text{val}_{\text{IP}}, d_{\text{max}}$  which will prove crucial to analysing IPGAP and **Greedy**. We begin by characterizing the value of the linear program:

► **Lemma 3.** *There exists  $c > 0$ , independent of  $n$ , such that with probability at least  $1 - \exp(-cn^{1-\delta})$ , we have that*

$$\frac{m}{d_{\text{max}}} \leq \text{val}_{\text{LP}} \lesssim \frac{1}{p}.$$

The proof is included in Appendix A, and follows from a maximum argument and a standard Chernoff bound. We note that the proof also implies  $\mathbb{P}(\text{IP is feasible}) \geq 1 - \exp(-cn^{1-\delta})$ . Although Lemma 3 readily yields  $\text{val}_{\text{IP}} \geq m/d_{\text{max}}$ , we highlight that this lower bound is not tight whenever  $mp \gg \log n$ . Indeed, we apply the first moment method to obtain a tighter lower bound on  $\text{val}_{\text{IP}}$  in this regime:

► **Lemma 4.** *Let  $mp \gg \log n$ . For any  $D \geq 1$  and  $n$  large enough, with probability at least  $1 - n^{-D}$  we have that*

$$\frac{1}{p} \log \left( \frac{mp}{\log n} \right) \lesssim \text{val}_{\text{IP}}$$

The proof of Lemma 4 is provided in Appendix A. Lemmas 3 and 4 come short of providing a full characterization of IPGAP, namely lacking an upper bound on  $\text{val}_{\text{IP}}$ . In this light, we turn our attention to the **Greedy** algorithm, and utilize it to construct a feasible integral solution and hence an upper bound on the value of IP. The analysis of **Greedy** crucially relies on characterizing the maximum inclusion set size,  $d_{\text{max}} := \max_{j \in [m]} |I_j|$ . The following lemma offers such a characterization in expectation, and evidences a key difference between the sparse and dense regimes of our problem:

---

**Algorithm 1** Greedy.

---

```

1:  $\mathcal{I} \leftarrow \{I_1, \dots, I_n\}$  ▷ Inclusion sets
2:  $U \leftarrow [m]$ 
3:  $t \leftarrow 0$ 
4: while  $|U| > 0$  do
5:    $P \leftarrow \operatorname{argmax}_{I \in \mathcal{I}} |I \cap U|$  ▷ Greedy step
6:    $\mathcal{I} \leftarrow \mathcal{I} \setminus \{P\}$ 
7:    $U \leftarrow U \setminus P$ 
8:    $t \leftarrow t + 1$ 
9:  $\operatorname{val}_{\text{Gr}} \leftarrow t$ 
10: return  $\operatorname{val}_{\text{Gr}}$ .
```

---

► **Lemma 5** (Maximum of Binomials). *Let  $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \text{Bin}(m, p)$ . Under the conditions in Assumption 2, it holds that*

$$\mathbb{E}d_{\max} = \mathbb{E} \max_{i \in [n]} X_i \sim \begin{cases} \frac{\log n}{\log(\log n / mp)} & , \text{ if } mp \ll \log n, \\ mp & , \text{ if } mp \gtrsim \log n. \end{cases}$$

The proof of Lemma 5 is provided in Appendix A, and involves a straight forward application of Markov's and Jensen's inequalities. Lemma 5 indicates a sharp transition between two regimes: the sparse regime  $mp \ll \log n$ , where binomial random variables are known to be well approximated by Poisson random variables, and the dense regime  $mp \gg \log n$ , where binomial random variables are known to be well approximated by Gaussian random variables. Importantly, in the sparse (Poisson-like) regime, the expected maximum of binomial random variables exceeds their individual expectations:  $\mathbb{E}X_1 \ll \mathbb{E}d_{\max}$ . Meanwhile in the dense (Gaussian-like) regime, the expected maximum and individual expectations are asymptotically equivalent up to multiplicative constants:  $\mathbb{E}X_1 \sim \mathbb{E}d_{\max}$ . This fine-grained characterization of the maxima of binomial random variables will prove essential to analysing the behaviour of **BlockGreedy** in Section 5. Finally, we characterize the asymptotic behaviour of  $d_{\max}$  and prove that  $d_{\max} \lesssim \mathbb{E}d_{\max}$  with high probability. Whilst this one sided result suffices for the forthcoming analysis, we expect a matching lower bound to hold as well. Additional insights into the concentration of  $d_{\max}$  may be found in Lemmas 19, 20, in Appendix A.

► **Lemma 6.** *Let  $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \text{Bin}(m, p)$ . Then, there exist constants  $c, \tilde{c} > 0$ , independent of  $n$ , such that*

$$\mathbb{P} \left( \max_{i \in [n]} X_i \geq c \cdot \mathbb{E} \max_{i \in [n]} X_i \right) \leq \frac{1}{n^{\tilde{c}}}.$$

The proof of Lemma 6 is provided in Appendix A.

## 5 Algorithmic solutions

### 5.1 Challenges of Greedy analysis

The aim of the present section is to conduct a rigorous analysis of the standard **Greedy** algorithm for the **Hitting Set** problem, within the prescribed Bernoulli random setting. In particular, we show that this routine succeeds at constructing hitting sets of optimal size w.h.p., as in the results of Section 4, up to multiplicative constants. This is done by first analysing a variation of the greedy heuristic, and subsequently proceeding by a reduction argument.

---

**Algorithm 2** BlockGreedy.
 

---

```

1: Let  $\mathcal{B}_t \subset \{I_1, \dots, I_n\}$  denote the  $t$ -th block, i.e. inclusion sets that become available at
   step  $t$ .
2:  $\mathcal{I} \leftarrow \emptyset$ 
3:  $U \leftarrow [m]$ 
4:  $t \leftarrow 0$ 
5: while  $|U| > 0$  and  $\mathcal{B}_t \neq \emptyset$  do
6:    $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{B}_t$  ▷ Adding elements from the new block
7:    $P \leftarrow \operatorname{argmax}_{I \in \mathcal{I}} |I \cap U|$  ▷ Greedy step
8:    $\mathcal{I} \leftarrow \mathcal{I} \setminus \{P\}$ 
9:    $U \leftarrow U \setminus P$ 
10:   $t \leftarrow t + 1$ 
11:  $\text{val}_{\text{BGr}} \leftarrow t$ 
12: if  $|U| > 0$  then cover the rest of  $U$  with a trivial algorithm,  $\text{val}_{\text{BGr}} \leftarrow \text{val}_{\text{BGr}} + |U|$ 
13: return  $\text{val}_{\text{BGr}}$ .

```

---

The core principle of **Greedy** is to construct a feasible solution in steps, by sequentially adding to the candidate solution an element which hits the largest number of remaining sets. In the chosen setting, where elements are added to sets with equal probability and independently of each other, we have precise estimates on the number of subsets hit by an element which is *picked first*. In fact, the size of this set is given by the maximum of independent Binomial random variables, which was analysed in Section 4. However, this very first step introduces nontrivial dependencies amongst the remaining matrix columns and significantly complicates keeping track of the marginal gains of each subsequent element addition to the candidate solution.

## 5.2 BlockGreedy algorithm

In order to circumvent this issue, we introduce a modified greedy routine, which we refer to as the **BlockGreedy** algorithm, where the elements of the ground set  $[n]$  are split into separate sets of a given size, which we call blocks. At the  $t$ -th iteration, the algorithm picks the element hitting the largest number of remaining sets across *the first  $t$  blocks only*. By choosing the size of the blocks appropriately, we have that at each iteration  $t$  one is guaranteed to find a solution of near-optimal size at least within the set of newly-included independent columns.

**BlockGreedy** is detailed in Algorithm 2, whilst informally, it works as follows.

1. Let  $K$  be the size of the solution (suggested by theoretical analysis);
2. Uniformly at random split  $n$  columns into  $K$  blocks with  $n/K$  columns per block;
3. Start with an empty set of possible choices of columns;
4. At the  $t$ -th iteration, first add the columns from the  $t$ -th block (Step 6). Then, perform one greedy step on the current set of possible choices (Step 7);
5. If after  $K$  iterations of the algorithm, some subsets remain uncovered, we use a trivial covering, i.e., covering each subset by a separate column.

Note that the first selection of the element which hits the most number of subsets again introduces dependencies. However, the columns that are in the newly added block are independent of everything else at time  $t$ . Let  $v_t$  be the element which is picked at the  $t$ -th step

of **BlockGreedy**,  $f_t$  be the number of new subsets that are hit by  $v_t^1$ , and  $F_t := \sum_{i=1}^t f_i$  be the total number of subsets which are hit after  $t$  steps. In order to analyse how many elements **BlockGreedy** has picked, we will consider the sequence  $f_1, f_2, \dots, f_s$ , with  $F_t := \sum_{i=1}^t f_i$ , such that the following holds:

1.  $F_s = m$ ;
2. if  $mp \lesssim \log n$ , then  $s \lesssim \text{val}_{\text{LP}}$ , otherwise,  $s \lesssim \text{val}_{\text{IP}}$ .

The first property ensures that **BlockGreedy** picks at most  $s$  elements, and the second property gives optimal bounds on  $s$ . One way to guarantee that **BlockGreedy** succeeds is to prove that among the choices of **BlockGreedy** at each step  $t$ , there was an element  $\tilde{v}_t$  which hits at least  $f_t$  new subsets w.h.p. We will prove that it is enough to look for  $\tilde{v}_t$  in the new block of columns  $\mathcal{B}_t$ , which are added at step  $t$ . Note that unless  $F_t = m$ , we have that  $f_t \geq 1$ , since each subset is hit by at least one element w.h.p.. Therefore, it will be enough to find a sequence  $\{f_1, f_2, \dots, f_v\}$  such that  $F_v \geq m - v$ , since it implies  $F_{2v} = m$ . This allows us to reduce the problem of proving the effectiveness of **BlockGreedy** to a key technical lemma. This lemma assumes that before step  $t$ , exactly  $F_{t-1}$  subsets are hit, and bounds from below the probability that some vertex in the new block will hit at least  $f_t$  new subsets. This boils down to computing  $\mathbb{P}(\text{Bin}(m - F_{t-1}, p) \geq f_t)$ .

► **Lemma 7** (Informal, see Lemma 26). *Let  $\varepsilon > 0$  and  $mp \lesssim \log n$ . For some constants  $\tau > 0$ ,  $1 < \alpha < \beta$ , and for  $t \in \mathbb{N}$ , let:*

$$f_t = \left\lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \right\rceil \quad \text{where } k \text{ is such that } \beta^{-k-1}m < m - F_{t-1} \leq \beta^{-k}m;$$

*Then there exists a choice of  $\tau, \alpha, \beta$  and  $K$ , such that  $F_K \geq m - K$  and  $K \sim \text{val}_{\text{LP}}$ . Furthermore, for this sequence  $f_t$  (which depends on  $\varepsilon$ ), for any  $t \leq K$ ,*

$$\mathbb{P}(\text{Bin}(m - F_{t-1}, p) \geq f_t) \geq n^{-\varepsilon}. \tag{5.1}$$

*Note that the implicit constants in the statements  $K \sim \text{val}_{\text{LP}}$  depend on  $\varepsilon$ .*

This lemma highlights the crucial dependency of the problem on the relationship between the average degree,  $mp$ , and  $\log n$ . For clarity of exposition, we only state the lemma for the case  $mp \lesssim \log n$  and refer to the Lemma 26 in the Appendix for the full version and corresponding proof. Here we comment on the intuition behind the proof.

When  $mp \lesssim \log n$ , we need to carefully track how the maximum degree changes. We look for an element which (i) covers a large number of subsets, i.e., close to the expected maximum number,  $\mathbb{E}d_{\max}$  and (ii) can be found with large enough probability. The second property is important for the reduction to the standard **Greedy** algorithm, whose direct analysis presents substantial difficulties, and is done later in this section. The quantity  $\mathbb{E}d_{\max}$  is sensitive to  $mp$  whenever the latter is close to  $\log n$ . Hence, we need to adjust which element we look for accordingly. This is done by setting  $f_t = \left\lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \right\rceil$  and increasing the parameter  $k$  as the number of remaining rows,  $m - F_t$ , decreases.

For example, consider the case  $mp = \log n$ . First, we can only pick a random element, since it will be as good (up to a multiplicative constant) as the maximal element. However, during the execution of the algorithm, the problem becomes more sparse, and if we continue to

<sup>1</sup> It may happen that  $v_t$  hits *more* than  $f_t$  new subsets. In this case, we still only count that exactly  $f_t$  are covered, and several extra sets will be covered multiple times in subsequent rounds. This overcounting simplifies the analysis and does not result in suboptimal solution.

### 30:10 Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem

pick random elements, we will construct a suboptimal solution. Therefore, we gradually increase how much the newly picked element will cover, *with respect to a random element*. This corresponds to the transition between Gaussian-like and Poisson-like behaviour of  $\text{Bin}(m - F_{t-1}, p)$ .

It is now straightforward to prove the following theorem, which makes rigorous the statements in Section 1.

► **Theorem 8.** *Under Assumption 2, we have that*

- (i) if  $mp \lesssim \log n$  then, for any  $\varepsilon > 0$  and  $n$  large enough,
- $$\mathbb{P}\left(\text{val}_{\text{BGr}} \lesssim \frac{m}{\mathbb{E}d_{\max}}\right) \geq 1 - \exp(-n^{1-\delta-\varepsilon});$$
- (ii) if  $mp \gg \log n$ , then, for any  $\varepsilon > 0$  and  $n$  large enough,
- $$\mathbb{P}\left(\text{val}_{\text{BGr}} \lesssim \frac{1}{p} \log\left(\frac{mp}{\log n}\right)\right) \geq 1 - \exp(-n^{1-\delta-\varepsilon}).$$
- (5.2)

Note that if  $mp \gtrsim n^\gamma$  for some  $\gamma > 0$ , then  $\log \frac{mp}{\log n} \sim \log n$ , and the bound in (ii) can be simplified.

**Proof.** The main idea of the proof is to analyse the distribution of the columns that are added at each step  $t$ . These columns are independent, and for each newly added column, the number of additional subsets which it covers is distributed according to  $\text{Bin}(m - F_{t-1}, p)$ , where  $F_{t-1}$  is the number of subsets which are already covered. Lemma 26 (see Lemma 7 above for an informal version) allows us to lower bound  $F_t$ , and we show now that we can do this with high probability.

Fix  $\varepsilon > 0$  and let  $\varepsilon' := \varepsilon/4$ . Let  $f_1, f_2, \dots$  be the sequence from Lemma 26 for  $\varepsilon'$  and let  $K$  be the value for which (C.1) is satisfied, i.e.  $F_K \geq m - K$ . Notice that  $K \leq C \max\left\{\frac{m}{\mathbb{E}d_{\max}}, \frac{1}{p} \log\left(\frac{mp}{\log n}\right)\right\}$  for some constant  $C > 0$ , for  $n$  large enough. We uniformly at random split  $n$  elements (columns) into  $K$  groups of size  $n/K$  each (assuming without loss of generality that  $K$  divides  $n$ , otherwise we consider groups of size  $\lfloor n/K \rfloor$ ), so that  $\mathcal{B}_t$  yields a new set of  $n/K$  elements at each iteration  $t \leq K$  and  $\mathcal{B}_t = \emptyset$  for  $t > K$ . We say that the algorithm fails at step  $t$  if before step  $t$ , at least  $F_{t-1}$  subsets are covered, but after step  $t$  less than  $F_t$  sets are covered. Using that, for  $n$  large enough, (i) columns in each newly added block are independent, (ii)  $\mathbb{P}(\text{Bin}(m - F_{t-1}, p) \geq f_t) \geq n^{-\varepsilon'}$ , and (iii)  $n/K \geq n^{1-\delta-\varepsilon'}$ , we get

$$\begin{aligned} \mathbb{P}(\text{BlockGreedy fails at step } t) &\stackrel{(i)}{\leq} (\mathbb{P}(\text{Bin}(m - F_{t-1}, p) < f_t))^{n/K} \\ &\stackrel{(ii)}{\leq} (1 - n^{-\varepsilon'})^{n/K} \\ &\stackrel{(iii)}{\leq} \exp(-n^{1-\delta-2\varepsilon'}). \end{aligned}$$

We then proceed by applying a union bound to obtain the result,

$$\begin{aligned} &\mathbb{P}(\text{BlockGreedy fails during first } K \text{ steps}) \\ &\leq \sum_{t=1}^K \mathbb{P}(\text{BlockGreedy fails at step } t) \leq K \cdot \exp(-n^{1-\delta-2\varepsilon'}) \leq \exp(-n^{1-\delta-3\varepsilon'}), \end{aligned}$$

where the second inequality holds since, by definition, the algorithm runs for  $K$  iterations, and the third one holds for  $n$  large enough. We proved that **BlockGreedy** succeeds in finding



at most  $K$  elements such that at most  $m - F_K$  sets remain uncovered. Since by construction,  $m - F_K \leq K$ , we can cover the remaining rows trivially using that IP is feasible by Lemma 16 with high probability, which proves that

$$\mathbb{P}(\text{val}_{\text{BGr}} \leq 2K) \geq 1 - \exp\left(-n^{1-\delta-4\varepsilon'}\right) = 1 - \exp\left(-n^{1-\delta-\varepsilon}\right),$$

for  $n$  large enough. Recalling that  $K \lesssim \text{val}_{\text{LP}}$  for  $mp \lesssim \log n$ , and that  $K \lesssim \text{val}_{\text{IP}}$  for  $mp \gg \log n$ , finishes the proof.  $\blacktriangleleft$

► **Corollary 9.** *Under Assumption 2, we have that for any  $D > 0$ ,*

(i) *for any  $n$  large enough,*

$$\mathbb{P}(\text{val}_{\text{BGr}} \sim \text{val}_{\text{IP}}) \geq 1 - n^{-D};$$

(ii) *if  $mp \lesssim \log n$ , then, for any  $n$  large enough,*

$$\mathbb{P}(\text{IPGAP} \sim 1) \geq 1 - n^{-D}; \tag{5.3}$$

(iii) *if  $mp \gg \log n$ , then, for any  $n$  large enough,*

$$\mathbb{P}\left(\text{IPGAP} \sim \log\left(\frac{mp}{\log n}\right)\right) \geq 1 - n^{-D}.$$

**Proof.** Proof follows from Lemma 3, Lemma 4, and Theorem 8.  $\blacktriangleleft$

### 5.3 Reduction from BlockGreedy to Greedy

With the above results at hand, we now proceed to analyse the **Greedy** algorithm by means of a suitable reduction. Recall that we denote outputs of **BlockGreedy** and **Greedy** as  $\text{val}_{\text{BGr}}$  and  $\text{val}_{\text{Gr}}$  respectively.

► **Theorem 10.** *Under Assumption 2 with  $\delta < 1/2$ , we have that, for  $n$  large enough,*

$$\mathbb{P}(\text{val}_{\text{Gr}} \sim \text{val}_{\text{IP}}) \geq 1 - \exp\left(-\sqrt{n}\right).$$

**Proof.** We use Theorem 8 with  $\varepsilon = 1/8 - \delta/4$ , and let  $K, \mathcal{B}_t$  be as defined in the proof of Theorem 8. We have that, for  $n$  large enough,

$$\mathbb{P}(\text{BlockGreedy fails at any step}) \leq \exp\left(-n^\Delta\right),$$

where  $\Delta := 3/4 - \delta/2 > 1/2$ .

Given a matrix  $\mathbf{A}$ , consider running the above definition of **BlockGreedy** for  $J := \exp(\sqrt{n})$  times, each time reshuffling the columns. In what follows, we address **BlockGreedy** and **Greedy** defined with the same tie-breaking strategy when it comes to a number of elements hitting the same number of sets, i.e., selecting the left-most column in the associated matrix  $\mathbf{A}$ . Both  $\text{val}_{\text{BGr}}$  and  $\text{val}_{\text{Gr}}$  are random variables, but conditioned on  $\mathbf{A}$ ,  $\text{val}_{\text{Gr}}$  is deterministic, while  $\text{val}_{\text{BGr}}$  still depends on the randomness of separating columns into blocks. Using the union bound, we have that

$$\begin{aligned} \mathbb{P}(\text{val}_{\text{Gr}} > 2K) &\leq \mathbb{P}(\exists \text{ a failed copy of BlockGreedy}) \\ &\quad + \mathbb{P}(\text{val}_{\text{BGr}} < \text{val}_{\text{Gr}} \text{ over all } J \text{ copies}). \end{aligned} \tag{5.4}$$

Applying the union bound again, we can upper bound the first term in (5.4):

$$\mathbb{P}(\exists \text{ a failed copy of BlockGreedy}) \leq J \exp\left(-n^\Delta\right) = \exp\left(-n^\Delta + n^{1/2}\right). \tag{5.5}$$

## 30:12 Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem

Now we focus on the second term in (5.4). Let  $v_1, v_2, \dots, v_g$  be the ordered sequence of elements picked by **Greedy**. Let  $M_t := \{v_1 \in \mathcal{B}_1, v_2 \in \mathcal{B}_1 \cup \mathcal{B}_2, \dots, v_t \in \mathcal{B}_1 \cup \dots \cup \mathcal{B}_t\}$ . The event  $\{\text{val}_{\text{BGr}} \geq \text{val}_{\text{Gr}}\}$  contains the event  $M_g$ , since in this case **BlockGreedy** will necessarily pick exactly the same columns  $v_1, v_2, \dots, v_g$ . Given that each reshuffling of the columns generates a uniform distribution of  $\mathcal{B}_i$ 's over possible partitions of  $n$  columns, we get that

$$\mathbb{P}(M_g) = \mathbb{P}(v_1 \in \mathcal{B}_1) \mathbb{P}(v_2 \in \mathcal{B}_1 \cup \mathcal{B}_2 \mid M_1) \dots \mathbb{P}(v_g \in \mathcal{B}_1 \cup \dots \cup \mathcal{B}_g \mid M_{g-1}).$$

The  $t$ -th term in the product above is equal to

$$\mathbb{P}(v_t \in \mathcal{B}_1 \cup \dots \cup \mathcal{B}_t \mid M_{t-1}) = \frac{t \frac{n}{K} - (t-1)}{n - (t-1)} \geq \frac{t}{K} - \frac{t-1}{n} \geq \frac{t}{2(K-1)},$$

where the last inequality holds for  $n \geq 4K$  (recall that  $n \gg K$ ). Since  $M_g \subset \{\text{val}_{\text{BGr}} \geq \text{val}_{\text{Gr}}\}$ , we can lower bound the probability of the latter event as follows (note that when  $g < K$  there will be less terms in the product, hence,  $\mathbb{P}(M_g)$  will be even larger),

$$\begin{aligned} \mathbb{P}(\text{val}_{\text{BGr}} \geq \text{val}_{\text{Gr}} \text{ for 1 copy}) &\geq \mathbb{P}(M_g) \\ &\geq \prod_{t=1}^{K-1} \mathbb{P}(v_t \in \mathcal{B}_1 \cup \dots \cup \mathcal{B}_t \mid M_{t-1}) \geq \prod_{t=1}^{K-1} \frac{t}{2(K-1)} \geq e^{-2K}, \end{aligned}$$

where we used that  $k! \geq (k/e)^k$  in the last inequality. Since  $K \leq C \max\left\{\frac{m}{\mathbb{E}d_{\max}}, \frac{1}{p} \log\left(\frac{mp}{\log n}\right)\right\}$  and  $1/p \leq n^\delta$ , there exists a constant  $\tilde{C} > 0$  large enough, such that  $K \leq \tilde{C} n^\delta \log n$ . Therefore, using independence of the reshuffling between the copies, we can compute

$$\begin{aligned} \mathbb{P}(\text{val}_{\text{BGr}} < \text{val}_{\text{Gr}} \text{ over all } J \text{ copies}) &= (1 - \mathbb{P}(\text{val}_{\text{BGr}} \geq \text{val}_{\text{Gr}} \text{ for 1 copy}))^J \\ &\leq (1 - e^{-2K})^J \\ &\leq \exp\left(-e^{\sqrt{n} - 2\tilde{C}n^\delta \log n}\right). \end{aligned} \tag{5.6}$$

Combining (5.4), (5.5) and (5.6), we showed that  $\mathbb{P}(\text{val}_{\text{Gr}} > 2K) \leq \exp(-\sqrt{n})$  for  $n$  large enough, which finishes the proof.  $\blacktriangleleft$

► **Remark 11.** We note that the  $\delta < 1/2$  condition in Theorem 10 is likely not optimal, and could be relaxed by reducing to **BlockGreedy** with more carefully chosen sets  $\mathcal{B}_t$ . In particular, the appropriate set sizes  $|\mathcal{B}_t|$  may not be identical across  $t \leq K$ . The analysis becomes more technical in this case, and we highlight this as an interesting open direction.

## 6 Discussion and Open Questions

Our work characterises multiplicative integrality gaps for the random hitting set problem. In this section, we discuss the intuition behind our main results, together with open questions and conjectures.

### 6.1 Summary of our results and proof techniques

We identified that the nature of integrality gaps depends on the size of the inclusion set, also viewed as the sparsity of the underlying hypergraph. In particular, when the average degree of a vertex is small, i.e., when each element belongs to a small number of subsets, we proved that there exists only a constant gap between linear and integer program solutions, together with a simple algorithmic solution. The situation changes when the hypergraph

becomes dense, where we show an increasing integrality gap. This separation stems mostly from the property of the binomial distribution, where the maximum of random variables grows identically to the expected value whenever the expected value is large, but is away from it if  $mp \ll \log n$ .

In our analysis of **BlockGreedy**, we track this change of behaviour using a geometric series, which means that the further we are in the execution of the algorithm, the larger the ratio between the element we pick and the average element will be. This picture coincides exactly with how the binomial distribution will behave if we decrease the average degree: for large instances, it will look approximately as a Gaussian, but when the average degree is small, Poisson approximation starts to dominate, the right tail becomes heavier, and the difference between  $d_{\max}$  and  $mp$  increases. Our analysis tracks the transition between Gaussian and Poisson-like behavior.

## 6.2 Multiplicative vs. additive integrality gaps

Our result only concerns multiplicative gaps, but the constants in our analysis can be large. This might be a consequence of the generality of the studied problem. For example, if one focuses only on the case of constant  $p$ , which immediately implies a very dense instance in our characterization, [16] proves that a simple algorithm is optimal for approximating the integer program up to a small additive error. Proving similar upper bounds on the constant in more general cases is an interesting open problem. Based on numerical experiments, we formulate the following conjectures.

► **Conjecture 12** (Very sparse). For  $mp \ll 1$ ,  $\frac{\text{val}_{\text{Gr}}}{\text{val}_{\text{LP}}} \rightarrow 1$ .

► **Conjecture 13** (Sparse). For  $1 \lesssim mp \ll \log n$ ,  $\frac{\text{val}_{\text{Gr}}}{\text{val}_{\text{IP}}} \rightarrow 1$ , and  $\frac{\text{val}_{\text{IP}}}{\text{val}_{\text{LP}}} \rightarrow C_1 \in (1, 1.5)$ .

► **Conjecture 14** (Dense). For  $mp \gg \log n$ ,  $\frac{\text{val}_{\text{Gr}}}{\text{val}_{\text{IP}}} \rightarrow C_2 \in (1, 1.5)$ .

## 6.3 Analysis of a linear program solution.

One motivation for studying the gaps between the integer and linear programs together with the solutions of linear programs themselves is to construct a rounding scheme which converts a fractional solution to an integer one. We believe this is another interesting direction for future work. In particular, numerical experiments show that entries which have large value in the fractional solution have a strong tendency to correspond to elements that are picked for the integer solution. This supports the claim that a combination of the greedy and linear programming approach might be fruitful in efficiently solving **Hitting Set**. One approach for further study consists of first solving a linear program, initializing  $\mathbf{x}$  with the largest elements in the linear solution, and greedily covering the remaining subsets.

---

### References

- 1 Gabriel Arpino, Daniil Dmitriev, and Nicolo Grometto. Greedy heuristics and linear relaxations for the random hitting set problem, 2023. [arXiv:2305.05565](https://arxiv.org/abs/2305.05565).
- 2 Sander Borst, Daniel Dadush, Sophie Huiberts, and Samarth Tiwari. On the integrality gap of binary integer programs with gaussian data. *Mathematical Programming*, 2022. doi:10.1007/s10107-022-01828-1.
- 3 Sander Borst, Daniel Dadush, and Dan Mikulincer. Integrality gaps for random integer programs via discrepancy. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2023. doi:10.1137/1.9781611977554.ch65.

- 4 Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*. World Scientific, 2000. doi:10.1142/4252.
- 5 Yaniv Erlich, Anna Gilbert, Hung Ngo, Atri Rudra, Nicolas Thierry-Mieg, Mary Wootters, Dina Zielinski, and Or Zuk. Biological screens from linear codes: theory and tools. *BioRxiv*, 2015. doi:10.1101/035352.
- 6 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM (JACM)*, 1998. doi:10.1145/285055.285059.
- 7 William Feller and Philip M Morse. *An introduction to probability theory and its applications*. American Institute of Physics, 1958. doi:10.1063/1.3062516.
- 8 Antonio Fernández Anta, Miguel A Mosteiro, and Jorge Ramón Muñoz. Unbounded contention resolution in multiple-access channels. *Algorithmica*, 2013. doi:10.1007/s00453-013-9816-x.
- 9 Abdolhossein Hoorfar and Mehdi Hassani. Inequalities on the lambert w function and hyperpower function. *Journal of Inequalities in Pure and Applied Mathematics*, 2008. URL: <https://arxiv.org/abs/2305.05565>.
- 10 Fotis Iliopoulos and Ilias Zadik. Group testing and local search: is there a computational-statistical gap? In *Conference on Learning Theory*. PMLR, 2021. URL: <https://proceedings.mlr.press/v134/iliopoulos21a.html>.
- 11 David S Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, 1973. doi:10.1145/800125.804034.
- 12 László Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 1975. doi:10.1016/0012-365X(75)90058-8.
- 13 Marc Mézard and Marco Tarzia. Statistical mechanics of the hitting set problem. *Physical Review E*, 2007. doi:10.1103/PhysRevE.76.041124.
- 14 Vangelis T Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys (CSUR)*, 1997. doi:10.1145/254180.254190.
- 15 Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996. doi:10.1145/237814.237991.
- 16 Orestis A Telelis and Vassilis Zissimopoulos. Absolute  $O(\log m)$  error in approximating random set covering: an average case analysis. *Information Processing Letters*, 2005. doi:10.1016/j.ipl.2005.02.009.
- 17 Ramon Van Handel. Probability in high dimension. Lecture notes, 2014. URL: <https://api.semanticscholar.org/CorpusID:124828412>.

## A Auxiliary lemmas

► **Lemma 15** (Lower Bound in Lemma 3). *We have that*

$$\text{val}_{LP} \geq \frac{m}{d_{\max}}.$$

**Proof.** Let  $\mathbf{x}_{LP}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_m^*)$  be an optimal solution for (1.2). Since  $A\mathbf{x}_{LP}^* \geq \mathbf{1}$  entrywise, by summing all entries we obtain that

$$m \leq \sum_i \mathbf{x}_i^* X_i \leq d_{\max} \sum_i \mathbf{x}_i^* = d_{\max} \text{val}_{LP}.$$

which upon rearranging yields the desired result. ◀

In addition to the above, we have the following elementary upper bound on  $\text{val}_{LP}$ , which holds both in the sparse and dense regime.

► **Lemma 16** (Upper Bound in Lemma 3). *There exists  $c > 0$ , independent of  $n$ , such that*

$$\mathbb{P}\left(\text{val}_{LP} \lesssim \frac{1}{p}\right) \geq 1 - \exp(-cn^{1-\delta}).$$

*This also implies that  $\mathbb{P}(\text{IP is feasible}) \geq 1 - \exp(-cn^{1-\delta})$ .*

**Proof.** Consider the candidate feasible solution  $\hat{\mathbf{x}} := \frac{1}{\tilde{C}np} \mathbf{1}$ , for some constant  $0 < \tilde{C} < 1$ . The following results from applying a union bound over constraints and the standard Chernoff bound.

$$\begin{aligned} \mathbb{P}(\hat{\mathbf{x}} \text{ not feasible}) &= \mathbb{P}(\exists i \in [m] : (\mathbf{A}\hat{\mathbf{x}})_i < 1) \\ &\leq m\mathbb{P}(\text{Bin}(n, p) < \tilde{C}np) \\ &\leq n^C \exp\left(-\frac{(1-\tilde{C})^2 np}{2}\right) \\ &\leq \exp(-cn^{1-\delta}). \end{aligned}$$

The desired conclusion follows by considering the complementary event to the one above and noting that  $\|\hat{\mathbf{x}}\|_1 \sim 1/p$ . Note that the event  $\{\hat{\mathbf{x}} \text{ is feasible for LP}\}$  implies the event  $\{\text{IP is feasible}\}$ .  $\blacktriangleleft$

► **Lemma 17** (Lambert  $W$  function, [9]). *For any  $x \geq e$ , there holds that*

$$\log x - \log \log x + \frac{\log \log x}{2 \log x} \leq W_0(x) \leq \log x - \log \log x + \frac{e}{e-1} \frac{\log \log x}{\log x}. \quad (\text{A.1})$$

In particular,

$$W_0(x) = \log x - \log \log x + o(1), \quad \text{as } x \rightarrow \infty. \quad (\text{A.2})$$

In addition, for any  $x \geq 1/e$ , the following identity is satisfied

$$W_0(x) = \log \frac{x}{W_0(x)}. \quad (\text{A.3})$$

**Proof of Lemma 4.** Fix  $D \geq 1$ . Let  $Z_k := |\{\mathbf{x} \in \{0, 1\}^m : \mathbf{A}\mathbf{x} \geq \mathbf{1}, \|\mathbf{x}\|_1 = k\}|$  be the number of feasible solutions of norm exactly  $k$ . Clearly,  $Z_k \leq Z_{k+1}$  for any  $k \geq 0$ . We also have that

$$\mathbb{E}Z_k = \sum_{\|\mathbf{x}\|_1=k} \mathbb{P}((\mathbf{A}\mathbf{x})_i \geq 1, \forall i \in [m]) = \binom{n}{k} (1 - (1-p)^k)^m.$$

We will now show that for  $k \ll \frac{1}{p} \log\left(\frac{mp}{\log n}\right)$ , we have  $\mathbb{E}Z_k \leq n^{-D}$ . Using that  $p \leq 1/2$  from Assumption 4 and that for  $x \in (0, \frac{1}{2})$ , we have  $(1-x)^y \geq e^{-2xy}$ , we can bound

$$\begin{aligned} \mathbb{E}Z_k &= \binom{n}{k} (1 - (1-p)^k)^m \leq n^k (1 - e^{-2pk})^m \\ &\leq n^k e^{-me^{-2pk}} = \exp\{k \log n - me^{-2pk}\}. \end{aligned}$$

Therefore,  $\mathbb{E}Z_k \leq n^{-D}$  will follow from

$$2pke^{2pk} \leq -2Dpe^{2pk} + \frac{2mp}{\log n}. \quad (\text{A.4})$$

Since  $k \ll \frac{1}{p} \log\left(\frac{mp}{\log n}\right)$ , we also have that  $k \leq k_* := \frac{1}{2p} W_0\left(\frac{mp}{D \log n}\right)$  for  $n$  large enough. For  $k = k_*$ , the left hand side of (A.4) is equal to  $\frac{mp}{D \log n}$ , while the right hand side is lower bounded by  $\frac{mp}{\log n}$ . Since  $D \geq 1$ , we recover that  $\mathbb{E}Z_k \leq n^{-D}$ . Note that for  $n$  large enough,  $\text{val}_{\text{IP}} \ll \frac{1}{p} \log \frac{mp}{\log n}$  implies that  $Z_{k_*} > 0$ . Therefore, applying Markov's inequality, we get that

$$\mathbb{P}\left(\text{val}_{\text{IP}} \ll \frac{1}{p} \log\left(\frac{mp}{\log n}\right)\right) \leq \mathbb{P}(Z_{k_*} > 0) \leq \mathbb{E}Z_{k_*} \leq n^{-D}, \quad (\text{A.5})$$

### 30:16 Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem

and the proof follows by considering the complementary events. Note that using similar derivations, one can also show that for  $k^* := \frac{1}{p} \log \left( \frac{1}{\delta} \frac{mp}{\log n} \right)$ , where  $\delta$  is defined in Assumption 4, we have  $\mathbb{E}Z_{k^*} \geq 1$ .  $\blacktriangleleft$

**Proof of Lemma 5.** For ease of notation, let us define  $b_n := \frac{\log n}{mp}$ ,  $b_n^* := \frac{1}{e}(b_n - 1)$ ,  $g_n := \frac{\log n}{\log(\log n/mp)}$ . We begin by proving the desired upper bound. By Jensen's inequality and bounding the maximum of positive values by their sum, for any  $\lambda > 0$ , we obtain

$$\mathbb{E} \max_{i \in [n]} X_i \leq \frac{1}{\lambda} \log \mathbb{E} \exp \left( \lambda \max_{i \in [n]} X_i \right) = \frac{1}{\lambda} \log \mathbb{E} \left( \max_{i \in [n]} \exp(\lambda X_i) \right) \leq \frac{1}{\lambda} \log \sum_{i \in [n]} \mathbb{E} \exp(\lambda X_i).$$

Finally, computing the moment generating function of binomial random variables, together with the inequality  $1 - x \leq e^{-x}$  yields

$$\mathbb{E} \max_{i \in [n]} X_i = \frac{\log n + m \log(1 - p(1 - e^\lambda))}{\lambda} \leq \frac{\log n - mp(1 - e^\lambda)}{\lambda}.$$

In the regime where  $mp \gtrsim \log n$ , we may choose  $\lambda > 0$  arbitrary, independent of  $n$ , from which it immediately follows that  $\mathbb{E} \max_{i \in [n]} X_i \lesssim mp$ .

For  $mp \ll \log n$ , we proceed by differentiating the last line in the above display and setting the resulting expression to zero. From this, we may choose  $\lambda$  as the solution of the following.

$$e^{\lambda-1}(\lambda - 1) = b_n^*$$

Under the present assumptions, this is expressed in terms of the Lambert W function as  $\lambda = 1 + W_0(b_n^*)$ , so that by (A.3), we obtain

$$\mathbb{E} \max_{i \in [n]} X_i \leq \frac{\log n \left( 1 - \frac{1}{b_n} + \frac{b_n^*}{b_n} \frac{e}{W_0(b_n^*)} \right)}{1 + W_0(b_n^*)} \sim g_n.$$

In the dense  $mp \gtrsim \log n$  regime, a matching lower bound is easily obtained by noting that  $\mathbb{E} \max_{i \in [n]} X_i \geq \mathbb{E}X_1 = mp$ .

To deal with the sparse regime, let  $\tau = 1/16$ . From Markov's inequality,

$$\mathbb{E} \max_{i \in [n]} X_i \geq \tau g_n \mathbb{P} \left( \max_{i \in [n]} X_i = \lceil \tau g_n \rceil \right) = \tau g_n (1 - (1 - \mathbb{P}(X_1 = \lceil \tau g_n \rceil))^n).$$

Hence, applying Lemma 25, for  $n$  large enough,

$$\mathbb{E} \max_{i \in [n]} X_i \geq \tau g_n \left( 1 - \left( 1 - n^{-1/2} \right)^n \right) \geq (\tau/2)g_n,$$

thus providing a matching lower bound for the sparse regime.

In the intermediate threshold regime  $mp \sim \log n$ , the average and maximum of  $X_i$ 's become of the same order, that is  $mp \sim \mathbb{E}d_{\max} \sim \log n$ . The smooth transition follows by noting that in this regime,  $b_n, b_n^*, W_0(b_n^*) \sim 1$ .  $\blacktriangleleft$

**► Lemma 18 (Chernoff Bound - upper tail).** *Let  $X_1, \dots, X_n$  be independent random variables taking values in  $\{0, 1\}$ ,  $X$  denote their sum and  $\mu = \mathbb{E}X$ . Then for any  $\delta > 0$ ,*

$$\mathbb{P}(X \geq (1 + \delta)\mu) \leq e^{-\delta^2 \mu / (2 + \delta)}.$$

In order to deal with concentration of  $d_{\max}$  around its expectation, we state the following useful result on tensorization of variance. We introduce notation  $\text{Var}_i$  and  $\mathbb{E}_i$ , where subscript  $i$  indicates conditioning on each component of an underlying random vector, except for the  $i$ -th one.

► **Lemma 19** (Theorem 2.3, [17]). *Let  $X_1, \dots, X_n$  be independent random variables and for each function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , define*

$$\text{Var}_i f(x_1, \dots, x_n) := \text{Var}(x_1, \dots, x_{i-1}, X_i, x_{i+1}, \dots, x_n).$$

*Then, there holds that*

$$\text{Var}(f(X_1, \dots, X_n)) \leq \mathbb{E} \sum_{i=1}^n \text{Var}_i f(X_1, \dots, X_n)$$

► **Lemma 20** (Concentration for  $d_{\max}$ ). *Let  $X_1, \dots, X_n \stackrel{\text{iid}}{\sim} \text{Bin}(m, p)$ . Then, for any  $t > 0$ ,*

$$\mathbb{P}(|d_{\max} - \mathbb{E}d_{\max}| > t) \leq \frac{mp}{t^2}.$$

► **Remark 21.** Note that in all regimes of  $m, p$  satisfying Assumption 2, choosing  $t \sim \mathbb{E}d_{\max}$  is sufficient to deduce from the previous lemma that  $d_{\max} \sim \mathbb{E}d_{\max}$  w.h.p..

**Proof.** Proceeding by Chebyshev's inequality, it suffices to show that  $\text{Var}(d_{\max}) \leq mp$ . By Lemma 19, we have that

$$\begin{aligned} \text{Var}(d_{\max}) &\leq \mathbb{E} \sum_{i=1}^n \mathbb{E}_i (d_{\max} - \mathbb{E}_i d_{\max})^2 \\ &= \mathbb{E} \sum_{i=1}^n \mathbb{E}_i \left[ (d_{\max} - \mathbb{E}_i d_{\max})^2 \mid d_{\max} = X_i \right] \mathbb{P}d_{\max} = X_i \\ &\quad + \mathbb{E} \sum_{i=1}^n \mathbb{E}_i \left[ (d_{\max} - \mathbb{E}_i d_{\max})^2 \mid d_{\max} \neq X_i \right] \mathbb{P}d_{\max} \neq X_i \\ &= \frac{1}{n} \mathbb{E} \sum_{i=1}^n \text{Var} X_i \\ &\leq mp, \end{aligned}$$

which is as required. ◀

**Proof of Lemma 6.** Let us consider the sparse and dense regimes separately.

In the dense regime for  $mp \gtrsim \log n$ , there exist constants  $c_1, c_2, c_3 > 0$  such that  $c_1 mp \leq \mathbb{E} \max_{i \in [n]} X_i \leq c_2 mp$ , as argued in Lemma 5, and  $mp \geq c_3 \log n$ . We apply the union and Chernoff bounds as in Lemma 18 to obtain, for any  $t \geq 1/c_1$ ,

$$\begin{aligned} \mathbb{P} \left( \max_{i \in [n]} X_i \geq t \cdot \mathbb{E} \max_{i \in [n]} X_i \right) &\leq n \mathbb{P}(X_1 \geq tc_1 mp) \\ &\leq n \exp \left( - \frac{(tc_1 - 1)^2 mp}{1 + tc_1} \right) \\ &\leq n \exp \left( - \frac{c_3 (tc_1 - 1)^2 \log n}{1 + tc_1} \right). \end{aligned}$$

It now suffices to choose  $t$  as a function of  $c_1, c_3$  such that  $\frac{c_3 (tc_1 - 1)^2}{1 + tc_1} > 1$ . By rearranging and solving the resulting quadratic equation, it follows immediately that any  $t > \frac{1}{c_1} + \frac{1 + \sqrt{1 + 8c_3}}{2c_3 c_1} > \frac{1}{c_1}$  suffices. Hence, there exist universal constants  $c, \bar{c}$ , such that the desired conclusion holds. We now consider the sparse regime  $mp \ll \log n$ , where by Lemma 5 there exists  $c_4 > 0$  such that  $mp \leq c_4 \log n / \log \left( \frac{\log n}{\log mp} \right)$ . Notice that for any  $\lambda > 0$ ,  $\max_{i \in [n]} X_i \leq \frac{1}{\lambda} \log \sum_{i=1}^n e^{\lambda X_i}$ . We apply Markov's inequality to obtain, for any  $t > 0$ ,

$$\begin{aligned}
 \mathbb{P}\left(\max_{i \in [n]} X_i \geq t \cdot \mathbb{E} \max_{i \in [n]} X_i\right) &\leq \mathbb{P}\left(\sum_{i=1}^n e^{\lambda X_i} \geq e^{\lambda t \mathbb{E} \max_{i \in [n]} X_i}\right) \\
 &\leq \frac{n \mathbb{E} e^{\lambda X_1}}{\exp(\lambda t \mathbb{E} \max_{i \in [n]} X_i)} \\
 &= \frac{n(1-p+pe^\lambda)^m}{\exp(\lambda t \mathbb{E} \max_{i \in [n]} X_i)} \\
 &\leq \exp\left(\log n + mp(e^\lambda - 1) - \frac{\lambda t c_4 \log n}{\log\left(\frac{\log n}{mp}\right)}\right),
 \end{aligned}$$

where we used that  $1+x < e^x$  to obtain the last inequality. Finally, by choosing  $t = 3/c_4$  and  $\lambda = \log(\log n/mp)$ , we obtain

$$\mathbb{P}\left(\max_{i \in [n]} X_i \geq \frac{3}{c_4} \cdot \mathbb{E} \max_{i \in [n]} X_i\right) \leq \frac{1}{n}. \quad \blacktriangleleft$$

► **Lemma 22** (Asymptotic expression for binomial probability mass function).

Let  $a \equiv a(n)$  and  $b \equiv b(n)$  be such that

1.  $1 \ll b \ll \sqrt{a}$ ,
2.  $p \ll 1$ .

If  $b \geq Cap$  for  $C > 1$ , then

$$\log \mathbb{P}(\text{Bin}(\lceil a \rceil, p) = \lceil b \rceil) \geq -\left(b \log \frac{b}{ap} - b + ap\right)(1 + o(1)), \quad (\text{A.6})$$

If also  $b \gg ap$ , we have that

$$\log \mathbb{P}(\text{Bin}(\lceil a \rceil, p) = \lceil b \rceil) \geq -\left(b \log \frac{b}{ap}\right)(1 + o(1)), \quad (\text{A.7})$$

Furthermore, all bounds remain valid upon replacing  $\lceil a \rceil$  to  $\lfloor a \rfloor$ .

**Proof.** We defer the proof of Lemma 22 to the extended version of this work found in [1]. ◀

► **Lemma 23** (Binomial Monotonicity). Let  $S_m \sim \text{Bin}(m, p)$ . Then for  $r \geq mp$ , we have that  $\mathbb{P}(S_m = r+1) \leq \mathbb{P}(S_m = r)$  and  $\mathbb{P}(S_{m-1} = r) \leq \mathbb{P}(S_m = r)$ .

**Proof.** The proof follows a similar argument as that presented in [7].

$$\begin{aligned}
 \frac{\mathbb{P}(S_m = r+1)}{\mathbb{P}(S_m = r)} &= \frac{\binom{m}{r+1} p^{r+1} (1-p)^{m-r-1}}{\binom{m}{r} p^r (1-p)^{m-r}} \\
 &= \frac{\frac{m!}{(r+1)!(m-r-1)!} p^{r+1} (1-p)^{m-r-1}}{\frac{m!}{r!(m-r)!} p^r (1-p)^{m-r}} \\
 &= \frac{(m-r)p}{(r+1)(1-p)} \leq 1.
 \end{aligned}$$

Similar arguments show that  $\mathbb{P}(S_{m-1} = r) \leq \mathbb{P}(S_m = r)$ . ◀



## B Main tool for the case $mp \lesssim \log n$ and Proof of Lemma 25

► **Lemma 24.** *If  $mp \lesssim \log n$ , then, for any  $\varepsilon > 0$ , there exist constants  $\tau > 0$  and  $1 < \alpha < \beta$ , such that, for  $k \lesssim \log n$  and for any  $\tilde{m}$ , satisfying  $\beta^{-k-1}m \leq \tilde{m} \leq \beta^{-k}m$ , for all  $n$  large enough,*

$$\mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil\right) \geq n^{-\varepsilon}.$$

**Proof.** The proof is essentially a careful application of Lemma 22. Let  $\tau, \alpha, \beta$  be constants to be fixed later and  $\tilde{m} = \lfloor \beta^{-k-1}m \rfloor$ . Depending on whether we have  $mp \ll \log n$  or  $mp \sim \log n$ , different terms will dominate the asymptotic expression from Lemma 22.

We start with the case  $mp \ll \log n$ . From Lemma 5, this implies that  $mp \ll \mathbb{E}d_{\max} \ll \log n$ . Here we can fix  $\alpha \equiv 2$  and  $\beta \equiv 3$ . Applying (A.7) for  $a = 3^{-k-1}m$  and  $b = (2/3)^k \tau \mathbb{E}d_{\max}$ , we have:

$$\log \mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (2/3)^k \tau \mathbb{E}d_{\max} \rceil\right) \geq -(2/3)^k \tau \mathbb{E}d_{\max} \log\left(\frac{2^k 3 \tau \mathbb{E}d_{\max}}{mp}\right) (1 + o(1)) \quad (\text{B.1})$$

Recall that our goal is to show  $\log \mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (2/3)^k \tau \mathbb{E}d_{\max} \rceil\right) \geq -\varepsilon \log n$ . We first show that there exists  $\tau > 0$  satisfying the following two inequalities:

$$\begin{aligned} \text{(i)} \quad & (2/3)^k \tau (\log 3 + k \log 2) \frac{\mathbb{E}d_{\max}}{\log n} \leq \frac{\varepsilon}{4}, \\ \text{(ii)} \quad & (2/3)^k \tau \frac{\mathbb{E}d_{\max}}{\log n} \log\left(\frac{\mathbb{E}d_{\max}}{mp}\right) \leq \frac{\varepsilon}{4}. \end{aligned} \quad (\text{B.2})$$

Indeed, since  $\mathbb{E}d_{\max} \ll \log n$  and  $k \ll (3/2)^k$ , inequality (i) will be satisfied for any  $\tau > 0$  for  $n$  large enough. For (ii) we need to use explicit bound for  $\mathbb{E}d_{\max}$ , in particular from Lemma 5 we know that there exists  $C > 0$ , such that  $\mathbb{E}d_{\max} \leq C \log n / (\log \log n - \log mp)$  for  $n$  large enough. Plugging this into (ii), we get for  $k = 0$ ,

$$\tau \frac{\mathbb{E}d_{\max}}{\log n} \log\left(\frac{\mathbb{E}d_{\max}}{mp}\right) \leq \frac{\tau C (\log C + \log \log n - \log(\log \log n - \log mp) - \log mp)}{\log \log n - \log mp} = \tau C + o(1). \quad (\text{B.3})$$

For  $\tau = \varepsilon/(8C)$ , (ii) holds for  $k = 0$  for  $n$  large enough. By increasing  $k$  we only decrease left hand side of (ii), therefore, the same value of  $\tau$  works for any  $k \geq 0$ .

Finally, by adding (i) and (ii) we showed that, for  $n$  large enough,

$$\log \mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (\alpha/2)^k \tau \mathbb{E}d_{\max} \rceil\right) \geq -\frac{\varepsilon}{2} \log n (1 + o(1)) > -\varepsilon \log n,$$

which finishes the proof for the case  $mp \ll \log n$ .

Now we focus on the case  $mp \sim \log n$ . Here we apply (A.6) for the values  $a = \beta^{-k-1}m$  and  $b = (\alpha/\beta)^k \tau \mathbb{E}d_{\max}$  keeping in mind the condition  $b \geq Cap$  with  $C > 1$ . We have

$$\begin{aligned} & \log \mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil\right) \\ & \geq -\left((\alpha/\beta)^k \tau \mathbb{E}d_{\max} \log\left(\frac{\beta \alpha^k \tau \mathbb{E}d_{\max}}{mp}\right) - (\alpha/\beta)^k \tau \mathbb{E}d_{\max} + \beta^{-k-1}mp\right) (1 + o(1)) \end{aligned}$$

We pick  $\tau = \gamma mp / \mathbb{E}d_{\max}$ , for some constant  $\gamma > 1$  to be specified later. Note that this way condition for applying (A.6),  $\frac{b}{ap} \geq C > 1$ , is satisfied since  $\frac{b}{ap} \geq \frac{\tau \mathbb{E}d_{\max}}{mp} = \gamma > 1$ . This simplifies the latter expression to the following:

$$\begin{aligned} & \log \mathbb{P} \left( \text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \gamma mp \rceil \right) \\ & \geq -mp \left( (\alpha/\beta)^k \gamma \log(\beta \gamma \alpha^k) - (\alpha/\beta)^k \gamma + \beta^{-k-1} \right) (1 + o(1)) \end{aligned}$$

Since in this regime we have  $mp \leq D \log n$  for some  $D > 0$ , for  $n$  large enough, it is enough to show

$$(\alpha/\beta)^k \gamma \log(\beta \gamma \alpha^k) - (\alpha/\beta)^k \gamma + \beta^{-k-1} \leq \varepsilon/(2D).$$

We first show that there exist constants  $1 < \alpha < \beta$  and  $\gamma > 1$ , depending on  $\varepsilon$  and  $D$ , satisfying the following two inequalities for any  $k \geq 0$ :

$$\begin{aligned} \text{(i)} \quad & (\alpha/\beta)^k \left( \gamma \log \beta \gamma - \gamma + \frac{1}{\alpha^k \beta} \right) \leq \frac{\varepsilon}{4D}, \\ \text{(ii)} \quad & (\alpha/\beta)^k k \log \alpha \leq \frac{\varepsilon}{4D}. \end{aligned}$$

Note that left hand side of (i) decreases as  $k$  increases, therefore, it is enough to look at  $k = 0$ . We need to show that there exist  $\beta, \gamma > 1$ , depending on  $\varepsilon, D$  such that

$$f(\beta, \gamma) := \gamma \log \beta \gamma - \gamma + \frac{1}{\beta} \leq \frac{\varepsilon}{4D}.$$

Note that  $\frac{\partial f}{\partial \beta} = \gamma/\beta - 1/\beta^2 > 0$  and  $\frac{\partial f}{\partial \gamma} = \log \beta \gamma > 0$  as long as  $\beta \gamma > 1$ . Since  $f(1, 1) = 0$ , we can find  $\beta, \gamma > 1$ , close enough to 1, such that  $f(\beta, \gamma) \leq \varepsilon/(4D)$ . We use these values of  $\beta$  and  $\gamma$  (or, equivalently,  $\tau$ ). Since  $k \ll (\beta/\alpha)^k$ , there exists  $\alpha \in (1, \beta)$ , such that (ii) holds. Summing (i) and (ii) shows that, for  $n$  large enough,

$$\log \mathbb{P} \left( \text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \gamma mp \rceil \right) \geq -\frac{\varepsilon mp}{2D} (1 + o(1)) \geq -\frac{\varepsilon \log n}{2} (1 + o(1)) \geq -\varepsilon \log n.$$

We proved that for  $mp \lesssim \log n$ , for any  $\varepsilon > 0$ , for  $n$  large enough, there exists  $\tau, \alpha, \beta$ , such that

$$\Pr \left( \text{Bin}(\lfloor \beta^{-k-1} m \rfloor, p) = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil \right) \geq n^{-\varepsilon}.$$

Since  $\beta^{-k-1} mp < \beta^{-k} mp < \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil$ , from binomial monotonicity, Lemma 23, we have that for any  $\tilde{m}$  such that  $\beta^{-k-1} m \leq \tilde{m} \leq \beta^{-k} m$ ,

$$\mathbb{P} \left( \text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil \right) \geq n^{-\varepsilon}.$$

In order to deal with the more delicate sparse regime throughout the paper where  $mp \ll \log n$ , we apply the following technical lemma.

► **Lemma 25.** For  $mp \ll \log n$ ,  $\varepsilon > 0$ , and  $n$  large enough, we have

$$\mathbb{P} \left( \text{Bin}(m, p) = \left\lceil \frac{\varepsilon}{8} \frac{\log n}{\log(\log n / mp)} \right\rceil \right) \geq n^{-\varepsilon}. \quad \blacktriangleleft$$

**Proof of Lemma 25.** We follow the argument in Lemma 24 with  $k = 0$  and  $\mathbb{E}d_{\max}$  replaced by  $\log n / (\log \log n - \log mp)$ . Note that in the proof of Lemma 24, in the case  $mp \ll \log n$ , we only used that  $mp \ll \mathbb{E}d_{\max} \ll \log n$  and  $\mathbb{E}d_{\max} \leq C \log n / (\log \log n - \log mp)$  for some  $C > 0$ . Since both these properties remain true upon replacing  $\mathbb{E}d_{\max}$  with  $\log n / (\log \log n - \log mp)$ , the proof follows. Since  $\tau = \varepsilon/(8C)$ , in the setting of Lemma 25, and  $C = 1$  in this argument, we pick  $\tau = \varepsilon/8$ . ◀

### C

 Lemma 26, formal version of Lemma 7

► **Lemma 26.** Let  $\varepsilon > 0$ . Consider the following choices of  $f_1, f_2, \dots$ :

(i) if  $mp \lesssim \log n$ , for some constants  $\tau > 0$  and  $1 < \alpha < \beta$ ,

$$f_t = \left\lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \right\rceil \quad \text{where } k \text{ is such that } \beta^{-k-1}m < m - F_{t-1} \leq \beta^{-k}m;$$

(ii) if  $mp \gg \log n$ , and  $\log mp \ll \log n$ ,

$$f_t = \lceil mp(1-p)^{t-1} \rceil \quad \text{if } t \leq t^* := \left\lceil \frac{1}{p} \log \left( \frac{mp}{\log n} \right) \right\rceil,$$

$$f_t = \tilde{f}_{t-t^*}, \quad \text{otherwise, where } \tilde{f}_t \text{ is the sequence from the case } mp \lesssim \log n;$$

(iii) otherwise, i.e., when  $\log mp \gtrsim \log n$ ,

$$f_t = \lceil mp(1-p)^{t-1} \rceil.$$

Then, there exists  $K$ , such that

- (i)  $F_K \geq m - K$ ;
  - (ii) if  $mp \lesssim \log n$ , then  $K \sim \text{val}_{LP}$ ;
  - if  $mp \gg \log n$ , then  $K \sim \text{val}_{IP}$ .
- (C.1)

Furthermore, for this sequence  $f_t$  (which depends on  $\varepsilon$ ), for any  $t \leq K$ ,

$$\mathbb{P}(\text{Bin}(m - F_{t-1}, p) \geq f_t) \geq n^{-\varepsilon}. \quad (\text{C.2})$$

Note that the implicit constants in the statements  $K \sim \text{val}_{LP}$  or  $K \sim \text{val}_{IP}$  depend on  $\varepsilon$ .

**Proof.** We proceed in the proof by first showing that there exists  $\tilde{K}$ , such that  $m - F_{\tilde{K}} \lesssim \tilde{K}$ , and then, by increasing  $\tilde{K}$  by a multiplicative factor, we find  $K$  such that  $m - F_K \leq K$ .

**Case  $mp \lesssim \log n$ .** From Lemma 24, there exist constants  $\tau > 0$ ,  $\alpha, \beta$  with  $1 < \alpha < \beta$ , such that, for any  $\tilde{m}$ , satisfying  $\beta^{-k-1}m \leq \tilde{m} \leq \beta^{-k}m$ , for all  $n$  large enough,

$$\mathbb{P}\left(\text{Bin}(\tilde{m}, p) = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil\right) \geq n^{-\varepsilon}.$$

Recall that in this case  $f_t = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil$ , where  $k$  is such that  $\beta^{-k-1}m \leq m - F_{t-1} \leq \beta^{-k}m$  and  $F_t = \sum_{s=1}^t f_s$ . From Lemma 24 we have that  $\mathbb{P}(\text{Bin}(m - F_{t-1}, p) = f_t) \geq n^{-\varepsilon}$ . Our goal is to prove that there exists  $s \lesssim \text{val}_{LP} \sim m/\mathbb{E}d_{\max}$ , such that  $m - F_s \lesssim s$ .

► **Lemma 27.** Let  $t^{(k)} := \frac{\beta-1}{\beta\tau} \frac{m}{\mathbb{E}d_{\max}} \alpha^{-k}$ .

$$\begin{aligned} \text{If } & m - F_{t-1} \leq \beta^{-k}m \\ \text{then } & m - F_{t+t^{(k)}-1} \leq \beta^{-k-1}m. \end{aligned}$$

Informally, if after  $t-1$  steps of *BlockGreedy*, at most  $\beta^{-k}m$  subsets are uncovered, then after  $t+t^{(k)}-1$  steps, at most  $\beta^{-k-1}m$  subsets remain uncovered.

**Proof.** Let  $s \geq t$ . As long as  $m - F_{s-1} > \beta^{-k-1}m$ , we will always have  $f_s = \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil$ . We proceed by contradiction. Assume that  $m - F_{t+t^{(k)}-1} > \beta^{-k-1}m$ . This implies that for all  $s \in [t-1, t+t^{(k)}-1]$ , we have  $f_s = f := \lceil (\alpha/\beta)^k \tau \mathbb{E}d_{\max} \rceil$ . Therefore,

$$F_{t+t^{(k)}-1} - F_{t-1} = t^{(k)} f \geq \frac{m(\beta-1)}{\beta^{k+1}} = \beta^{-k}m - \beta^{-k-1}m,$$

## 30:22 Greedy Heuristics and Linear Relaxations for the Random Hitting Set Problem

and

$$\begin{aligned} m - F_{t+t^{(k)}-1} &= m - F_{t-1} - (F_{t+t^{(k)}-1} - F_{t-1}) \\ &\leq \beta^{-k}m - (\beta^{-k}m - \beta^{-k-1}m) = \beta^{-k-1}m. \end{aligned}$$

Therefore, we must have  $m - F_{t+t^{(k)}-1} \leq \beta^{-k-1}m$ .  $\blacktriangleleft$

Note that we always have  $\beta^{-1}m \leq m - F_0 = m$ . If we consecutively apply Lemma 27 starting with  $k = 0$ , then, for  $v(k) := \sum_{s=0}^k t^{(s)}$  we have  $m - F_{v(k)-1} \leq \beta^{-k-1}m$ . Therefore, for  $k := \frac{\log \mathbb{E}d_{\max}}{\log \beta}$ , we have  $m - F_{v(k)-1} \leq \frac{m}{\mathbb{E}d_{\max}}$ . We can bound

$$v(k) \leq \sum_{s=0}^{\infty} t^{(s)} = \frac{\beta - 1}{\beta\tau(\alpha - 1)} \frac{m}{\mathbb{E}d_{\max}} \sim \frac{m}{\mathbb{E}d_{\max}}.$$

From Lemma 6 we have  $d_{\max} \lesssim \mathbb{E}d_{\max}$  with high probability. Together with Lemma 15 this implies  $\text{val}_{\text{LP}} \geq \frac{m}{d_{\max}} \gtrsim \frac{m}{\mathbb{E}d_{\max}}$ . Now, if we pick  $\tilde{K} := v(k) \lesssim \frac{m}{\mathbb{E}d_{\max}}$ , we have that  $\text{val}_{\text{BGr}} \lesssim \frac{m}{\mathbb{E}d_{\max}}$ . Since  $\text{val}_{\text{LP}} \leq \text{val}_{\text{BGr}}$ , we have that  $\tilde{K} \sim \text{val}_{\text{LP}}$  and  $m - F_{\tilde{K}} \lesssim \tilde{K}$ .



**Case  $mp \gg \log n$ .** Here, we have that  $\mathbb{E}d_{\max} = mp(1 + o(1))$ , therefore, picking an element that hits an average number of subsets is approximately the same as picking an element that hits close to maximum number of subsets. From the properties of the mean and the median of the binomial distribution, it follows that  $\mathbb{P}(\text{Bin}(\tilde{m}, p) \geq \lceil \tilde{m}p \rceil) \geq 1/3$ , for any  $\tilde{m}$ .

We begin with the case  $\log mp \ll \log n$ . This means that  $mp$  cannot grow polynomially in  $n$ , but e.g.  $mp \sim \log^2 n$  is possible. In this regime,  $\text{val}_{\text{IP}} \sim \frac{1}{p} \log \left( \frac{mp}{\log n} \right)$ . Let  $K_1 = \left\lceil \frac{1}{p} \log \left( \frac{mp}{\log n} \right) \right\rceil$  and  $f_1, \dots, f_{K_1}$  be a sequence such that  $f_s = \lceil mp(1-p)^s \rceil$ . Then, we have that  $m - F_{K_1} \leq m(1-p)^{K_1} \leq \frac{1}{p} \log n$ . Therefore,  $(m - F_{K_1})p \sim \log n$ , and we can continue with  $\tilde{f}_t$  from the previous section  $mp \sim \log n$ , with  $\tilde{F}_t := \sum_{s=1}^t \tilde{f}_s$ . For this sequence  $\tilde{f}_1, \dots, \tilde{f}_{K_2}$ , we have  $K_2 \lesssim \frac{1}{p}$ , and  $m - F_{K_1} - \tilde{F}_{K_2} \lesssim \frac{1}{p} \ll \frac{1}{p} \log \left( \frac{mp}{\log n} \right)$ . The required statement holds for combined sequences  $f_t$  and  $\tilde{f}_t$  and  $\tilde{K} := K_1 + K_2$ .


Finally, we study the case  $\log mp \gtrsim \log n$ , which implies that  $\text{val}_{\text{IP}} \sim \frac{1}{p} \log n$ . This case is trivial, as one can pick  $\tilde{K} = \left\lceil \frac{1}{p} \log \left( \frac{mp}{\log n} \right) \right\rceil \lesssim \text{val}_{\text{IP}}$  and  $f_1, \dots, f_{\tilde{K}}$  a sequence such that  $f_s = \lceil mp(1-p)^s \rceil$ . Then, we have that  $m - F_{\tilde{K}} \leq m(1-p)^{\tilde{K}} \leq \frac{1}{p} \log n \lesssim \text{val}_{\text{IP}}$ .

**From  $m - F_{\tilde{K}} \lesssim \tilde{K}$  to  $m - F_K \leq K$ .** Finally, using that  $f_t \geq 1$  by Lemma 16 unless  $F_t = m$ , there exists some constant  $C > 0$ , such that for  $K := C\tilde{K}$ ,  $F_K \geq m - K$ , which finishes the proof.  $\blacktriangleleft$

# The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

Amnon Ta-Shma  

Department of Computer Science, Tel Aviv University, Israel

Ron Zadacario  

Department of Computer Science, Tel Aviv University, Israel

---

## Abstract

Numerous works have studied the probability that a length  $t - 1$  random walk on an expander is confined to a given rectangle  $S_1 \times \dots \times S_t$ , providing both upper and lower bounds for this probability. However, when the densities of the sets  $S_i$  may depend on the walk length (e.g., when all sets are equal and the density is  $1 - 1/t$ ), the currently best known upper and lower bounds are very far from each other. We give an improved confinement lower bound that almost matches the upper bound.

We also study the more general question, of how well random walks fool various classes of test functions. Recently, Golowich and Vadhan proved that random walks on  $\lambda$ -expanders fool *Boolean, symmetric* functions up to a  $O(\lambda)$  error in *total variation distance*, with *no dependence on the labeling bias*. Our techniques extend this result to cases not covered by it, e.g., to functions testing confinement to  $S_1 \times \dots \times S_t$ , where each set  $S_i$  either has density  $\rho$  or  $1 - \rho$ , for arbitrary  $\rho$ .

Technique-wise, we extend Beck’s framework for analyzing what is often referred to as the “flow” of linear operators, reducing it to bounding the entries of a product of  $2 \times 2$  matrices.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains; Theory of computation  $\rightarrow$  Expander graphs and randomness extractors

**Keywords and phrases** Expander random walks, Expander hitting property

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.31

**Category** RANDOM

**Related Version** *Full Version:* <https://ecc.weizmann.ac.il/report/2024/118> [14]

**Funding** *Amnon Ta-Shma:* The research leading to these results has received funding from the Israel Science Foundation (grant number 443/22)

*Ron Zadacario:* The research leading to these results has received funding from the Israel Science Foundation (grant number 443/22) and the Blavatnik Family Foundation.

**Acknowledgements** We thank the RANDOM 2024 anonymous referees for their helpful comments.

## 1 Introduction

Fix a set of vertices  $V = [n]$  and  $t$  subsets  $S_1, \dots, S_t \subseteq V$ . The hitting property of expander graphs [1] says that for a sufficiently good expander graph  $G$  on the set of vertices  $V$ , the probability that for all  $i = 1, \dots, t$  the  $i$ ’th step of a random walk on  $G$  falls inside  $S_i$  is small, and therefore, with a good probability, the walk escapes the confinement  $S_1 \times \dots \times S_t$ . Specifically,

► **Theorem 1** (Expander Hitting Property, based on [10]). *Let  $G = (V, E)$  be a  $\lambda$ -expander. Then, for every sequence of subsets  $S_1, \dots, S_t \subseteq V$  such that  $S_i$  is of density  $\rho_i = |S_i| / |V|$ ,*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] \leq \sqrt{\rho_1 \rho_t} \cdot \prod_{i=1}^{t-1} ((1 - \lambda) \sqrt{\rho_i \rho_{i+1}} + \lambda). \quad (1)$$



© Amnon Ta-Shma and Ron Zadacario;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 31; pp. 31:1–31:22



Leibniz International Proceedings in Informatics

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

## 31:2 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

We remark that a slightly weaker bound of  $\prod_{i=1}^{t-1} (\sqrt{\rho_i \rho_{i+1}} + \lambda)$  appears in [10]. For the case where all densities  $\rho_i$  are the same  $\rho$ , a bound of  $\rho((1-\lambda)\rho + \lambda)^{t-1}$  appears in [15], and of  $\rho(\rho + \lambda)^{t-1}$  appears in [2]. The bound in the general case (Equation 1) follows by a similar proof, with a slightly more careful analysis. See Subsection 4.1.

However, on a conceptual level, one expects an expander random walk to mimic a truly random walk, each time choosing a vertex uniformly at random independent of all other choices. I.e., ideally, we would have liked a bound stating that the probability of an expander random walk being confined to  $S_1 \times \dots \times S_t$  is roughly the same as the probability of the same event with respect to a walk on the complete graph with self loops (which equals the product of the densities of the sets). Indeed, for the case in which all densities are equal, the following has been proven in [2]:

- **Theorem 2** ([2]). *Let  $G = (V, E)$  be a  $\lambda$ -expander. For every sequence of subsets  $S_1, \dots, S_t \subseteq V$  such that  $S_i$  is of density  $\rho$ ,*
- *If  $\lambda < \rho/6$ , then  $\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] \geq \rho \cdot (\rho - 2\lambda)^{t-1}$ .*
  - *If  $\lambda < \rho^2/2$ , then  $\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] \geq \rho \cdot (\rho - \lambda)^{t-1}$ .*

How tight are these bounds?

To get a feeling for the upper and lower bounds, let us look at the special case where all densities  $\rho_i$  are the same  $\rho$ . In this case, independent sampling gives the exact answer  $\rho^t$ . The upper bound (Theorem 1) is  $\rho\mu^{t-1}$ , where  $\mu = \rho + (1-\rho)\lambda$ , and

$$|\rho\mu^{t-1} - \rho^t| = \rho(\mu^{t-1} - \rho^{t-1}) = \rho(\mu - \rho) \sum_{j=0}^{t-2} \rho^j \mu^{t-2-j} \leq \rho(1-\rho)\lambda \sum_{j=0}^{t-2} \rho^j \leq \rho \cdot \lambda,$$

where the first equality is because  $\mu \geq \rho$  and the second equality is using  $a^k - b^k = (a-b) \sum_{j=0}^{k-1} a^j b^{k-1-j}$ . We also use  $\mu - \rho = (1-\rho)\lambda$ . In particular the error term is at most  $\lambda$ , and tends to zero when  $\lambda$  tends to 0.

However, for the lower bound (Theorem 2), for any  $\lambda$  we have

$$\begin{aligned} |\rho^t - \rho(\rho - \lambda)^{t-1}| &= \rho(\rho^{t-1} - (\rho - \lambda)^{t-1}) = \rho\lambda \sum_{j=0}^{t-2} \rho^{t-2-j} (\rho - \lambda)^j \\ &\geq \rho\lambda \rho^{t-2} \sum_{j=0}^{t-2} (\rho - \lambda)^j \approx \rho^{t-1} \frac{\lambda}{\lambda + \frac{1}{t}}. \end{aligned}$$

Thus, when  $\lambda$  is some small constant, independent of  $t$  and  $\rho = 1 - 1/t$ , the difference between independent sampling and the lower bound is  $\rho^{t-1} \frac{\lambda}{\lambda + 1/t} \approx 1/e$ . Therefore, even for arbitrarily small  $\lambda$ , if we let  $t$  grow to infinity and we let the density  $\rho$  depend on  $t$ , there is a constant gap between the independent sampling probability and the lower bound! Thus, a natural question is: can we find a better lower bound that matches the independent probability? In this work we prove:

- **Theorem 3** (New confinement lower-bound). *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be each of density  $\rho$  for some  $\rho^1$ . If  $\lambda \leq \frac{\rho^2}{3}$ , then*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] \geq \rho \cdot (\rho - \lambda(1 - \rho^2))^{t-1}.$$

<sup>1</sup> In fact, we prove the theorem under more general conditions, see Section 4

This bound is close to the independent sampling probability:

$$\begin{aligned} |\rho^t - \rho \cdot (\rho - \lambda(1 - \rho^2))^{t-1}| &= \rho^t - \rho \cdot (\rho - \lambda(1 - \rho^2))^{t-1} \\ &= \rho \cdot \lambda(1 - \rho^2) \sum_{j=0}^{t-2} \rho^j (\rho - \lambda(1 - \rho^2))^{t-2-j} \\ &\leq \lambda \cdot \rho(1 - \rho^2) \cdot \sum_{j=0}^{\infty} \rho^j = \lambda \cdot \rho(1 + \rho) \leq 2\rho\lambda. \end{aligned}$$

Therefore, for any  $\lambda$ , if we let  $t$  grow to infinity, and even if we let the density  $\rho$  depend on  $t$ , the distance between the independent probability ( $\rho^t$ ) and the lower bound is at most  $2\lambda$  (instead of an absolute constant before).

## 1.1 Further Results

Expander random walks are typically used as a randomness-efficient way of generating a uniform-like sequence of vertices  $v_1, \dots, v_t$ . In most applications, the walk is used to “fool” a test function  $f$ . For example, we may think of the confinement problem when all sets  $S_i$  are the same set  $S$ , as taking an expander with  $|V|$  vertices, which we label with 0 or 1 according to membership in  $S$ . We set  $f$  to be the AND function. We compare the probability that  $f(x_1, \dots, x_t)$  evaluates to 1 when  $x_1, \dots, x_t$  are the labels obtained from a random walk on the graph (which is the quantity we want to bound) with the probability that  $f$  evaluates to 1 when the labels are obtained from vertices chosen uniformly at random (which is a known quantity and equals the density  $S$  raised to the power of  $t$ ). We wish to claim these two quantities are close to each other.

More generally, we say a test function  $f : \mathbb{Z}_{d'}^t \rightarrow \mathbb{Z}_d$  is  $\varepsilon$ -fooled by expander random walks if for every  $\lambda$ -expander graph  $G = (V, E)$  and every labeling  $\text{val} : V \rightarrow \mathbb{Z}_{d'}$ ,  $d_{TV}(f(\text{val}(\text{RW}_G^t)), f(\text{val}(\text{Ind}_V^t))) \leq \varepsilon$ , where

- $\text{RW}_G^t$  is the distribution obtained by taking a length  $t - 1$  random walk on  $G$ . That is, we sample  $v_1 \in V$  uniformly at random. Then, for  $i = 2, \dots, t$  sample  $v_i$  uniformly at random from the neighbours of  $v_{i-1}$ .  $f(\text{val}(\text{RW}_G^t))$  is the distribution of  $f(\text{val}(v_1), \dots, \text{val}(v_t))$  when  $(v_1, \dots, v_t)$  is sampled from  $\text{RW}_G^t$ .
- $\text{Ind}_V^t$  is the distribution obtained by sampling  $v_1, \dots, v_t \in V$  uniformly at random. Note that  $\text{Ind}_V^t = \text{RW}_J^t$  where  $J$  is the complete graph on  $V$  with self loops.  $f(\text{val}(\text{Ind}_V^t))$  is the distribution of  $f(\text{val}(v_1), \dots, \text{val}(v_t))$  when  $(v_1, \dots, v_t)$  is sampled from  $\text{Ind}_V^t$ .

Cohen et al. [4] proved that all Boolean *symmetric* functions  $f$  are fooled by expander random walks with up to a  $O(\lambda/\sqrt{\rho_{\min}})$  error in total variation distance, where  $\rho_{\min} = \min\{\rho_0, \rho_1\}$ , and  $\rho_b$  is the *density* of  $b$ , i.e., that fraction of vertices with label  $b$ . Thus, even in the symmetric Boolean case, the error bound of [4] is  $O(\lambda)$  only when  $\rho_{\min}$  is bounded from below by some constant. When  $\rho_{\min}$  is allowed to depend on  $t$ , the error bound of [4] may weaken as  $t$  increases.

A remarkable recent result of Golowich and Vadhan [8] significantly strengthened and extended the results of [4], and using new techniques managed to eliminate the dependence on the bias. That is, they prove that all symmetric *Boolean* functions are fooled by expander random walks with up to  $O(\lambda)$  error in total variation distance, where the constant hidden in the Big-O notation is absolute and does not depend on  $\rho_{\min}$ .

Notice that [8] implies that for confinement to a single set (which is a symmetric function) the difference between independent sampling and RW sampling is bounded by  $O(\lambda)$ , even when the density  $\rho$  may depend on  $t$ . Thus, it implies that Theorem 2, which gives constant

## 31:4 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

difference for  $\rho = 1 - 1/t$ , is not tight. In this regard, Theorem 3 gives a bound that replaces the  $O(\lambda)$  difference guaranteed by [8] with a more precise bound (that is in particular at most  $2\lambda$ ).

Let us now discuss whether there are functions for which the [8] bound does not guarantee an  $O(\lambda)$  error, while our technique does.

A first candidate for such a problem is the confinement problem for  $S_1 \times \dots \times S_t$ , where the sets  $S_i$  might be different, and are only guaranteed to all have the same density. Theorem 3 still guarantees the same bound, whereas [8] seems to not apply, because the function is not symmetric anymore. However, the Golowich-Vadhan result might be modified to cover this case as well, by using one fixed set, and adding corresponding permutation operators to the expanders, making them directed (which is still fine for [8]).<sup>2</sup>

However, using our techniques, we prove the following. Let  $\mathbf{1}_S(i)$  equal 1 if  $i \in S$  and 0 otherwise. Then,  $\mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}$  equals one if the input is confined to  $S_1 \times \dots \times S_t$  and zero otherwise. We prove:

► **Theorem 4.** *Let  $G = (V, E)$  be a  $\lambda$ -expander where  $\lambda \leq 1/3$ , and  $t \geq 1$  an integer. Let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets such that the largest subset also has the maximal variance. Then,*

$$d_{TV}(\mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{RW}_G^t), \mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{Ind}_V^t)) < 3\rho_{\max} \cdot \lambda,$$

where  $\rho_{\max}$  is the density of the largest subset.

In particular,

► **Corollary 5.** *Let  $G = (V, E)$  be a  $\lambda$ -expander where  $\lambda \leq 1/3$ , and  $t \geq 1$  an integer. Let  $S \subseteq V$  be a subset of density  $\rho$ , and suppose  $S_1, \dots, S_t \subseteq V$  are subsets such that for every  $i$ ,  $S_i = S$  or  $S_i = \bar{S}$ . Then,*

$$d_{TV}(\mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{RW}_G^t), \mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{Ind}_V^t)) < 3\rho \cdot \lambda.$$

Notice that these functions are not symmetric, and therefore the results of [8] do not apply to them, while our techniques still work.

We also use similar techniques to analyze the extent to which the sum function modulo  $d$  is fooled by expander random walks on graphs with arbitrarily biased labelings, and prove that it is fooled with an  $O(\sqrt{d} \cdot \lambda)$  error in total variation distance, with no dependence on the labeling bias. We prove:

► **Theorem 6.** *For integers  $t \geq 1$ ,  $d' \geq 2$ , and  $d \geq 2$  let  $G = (V, E)$  be a  $\lambda$ -expander where  $\lambda \leq 1/6$ . Let  $\text{val} : V \rightarrow \mathbb{Z}_{d'}$  be any labeling. Then*

$$d_{TV}(\text{Sum}_d(\text{val}(\text{RW}_G^t)), \text{Sum}_d(\text{val}(\text{Ind}_V^t))) \leq 5\sqrt{d} \cdot \lambda.$$

The  $O(\sqrt{d} \cdot \lambda)$  error term also follows from the work of [8] on width- $d$  permutation branching programs, using different techniques.

Additionally, we prove a bound on the bias of a labeling in terms of the density of the most frequent label,  $\rho_{\max}$ . This is in contrast to previous bias-dependent result (e.g [8] for symmetric functions over  $\mathbb{Z}_d$  with  $d > 2$ ) where the total variation bound degrades with  $\rho_{\min}$ , rather than  $1 - \rho_{\max}$  (and notice that always  $\rho_{\min} \leq 1 - \rho_{\max}$ ). This dependence is more

---

<sup>2</sup> We thank the anonymous referee for bringing this to our attention.



resilient as it can tolerate very rare labels, as long as the most common label is not too dominant. We think that this observation could potentially serve as an incentive to shift the bias dependence in previous works from the smallest label weight to the largest. Specifically, we prove,

► **Proposition 7.** *For a prime  $p$ , let  $\text{val} : [n] \rightarrow \mathbb{Z}_p$  be a labeling that assigns label  $a \in \mathbb{Z}_p$  to  $\rho_a$  fraction of the vertices, and denote  $\rho_{\max} = \max_a \rho_a$ . Then, for every non-trivial character  $\chi$  of  $\mathbb{Z}_p$ ,  $\text{bias}_\chi(\text{val}) \leq \sqrt{1 - \left(1 - \cos \frac{2\pi}{p}\right) (1 - \rho_{\max})}$ , where  $\text{bias}_\chi(\text{val}) \stackrel{\text{def}}{=} \left| \mathbb{E}_{i \in [n]} \chi(\text{val}(i)) \right|$ .*

We also point out that our proofs apply even if the graph is different for each of the  $t$  steps, as long as it is a  $\lambda$ -expander at each step. The same property holds in previous works as well, e.g [8].

## 1.2 The Technique

We extend the techniques of Gillman [6], Healy [9] and Beck [3], that established a framework for analyzing what is often referred to as the “flow” of linear operators. The flow of a linear operator  $T$  from the linear subspace  $V_2$  to the linear subspace  $V_1$  is the quantity  $\|\Pi_1 T \Pi_2\|$  where  $\Pi_i$  is the projection operator onto  $V_i$ . In our context,  $V_1$  and  $V_2$  will be either the line spanned by the all-ones vector (The “parallel space”), or its orthogonal complement (The “perpendicular space”).

Let  $G$  also denote the transition matrix of a  $\lambda$ -expander graph, and let  $P$  denote the projection matrix on the set  $S$ . That is,  $P$  is the diagonal matrix satisfying  $P[v, v] = 1$  if  $v \in S$  and 0 otherwise. The probability that a length  $t$  random walk on  $G$  never escapes  $S$  can be expressed algebraically as  $\mathbf{1}^T (PG)^{t-1} P \mathbf{1}$ , where we denote  $\mathbf{1} = \frac{1}{\sqrt{|V|}} (1, \dots, 1)^T$ .

One way to analyze this expression is to decompose the probability distribution at each of the  $t$  steps to its parallel and perpendicular components. The parallel component is identical to the independent sampling case, while the perpendicular component is shrunk by a factor of  $\lambda$  after each step on  $G$ . The above approach underlies many results in the field, and, in particular, the expander Chernoff bound [6, 9]. Beck [3] simplified the analysis by defining a  $2 \times 2$  “flow” matrix for a linear operator  $T$ . The  $i, j$ ’th entry of the flow matrix is the flow of  $T$  from  $V_j$  to  $V_i$ , where  $V_i$  and  $V_j$  are either the perpendicular space or the parallel space. This notation reduced the problem of bounding quantities like  $|\mathbf{1}^T T \mathbf{1}|$  to bounding the  $[0, 0]$  entry of a  $2 \times 2$  matrix with non-negative entries. In this language, the expression  $\mathbf{1}^T (PG)^{t-1} P \mathbf{1}$  is the flow of the operator  $(PG)^{t-1} P$  from the parallel space to itself. For more details about the flow framework see Section 3.

[2] proved their confinement probability lower bound by giving simultaneous upper and lower bounds on flows between the perpendicular and parallel spaces. However, they did it explicitly and specifically for the confinement problem with equal density at each step, and obtained sub-optimal bounds. In this paper we analyze flows emerging from confinement problems (and additional problems) using the  $2 \times 2$  flow matrix notation. As a result, we achieve simpler terms that are easier to follow and generalize to a broader setting of confinement problems with varying densities. These terms also indicate how to improve upon previous work (even when all densities are equal).

## 1.3 Summary and Discussion

As mentioned before, several total variation bounds in previous works depend on the labeling bias, namely on the weights  $\rho_b$  that are induced by a labeling. Cohen et al. [4] proved that all Boolean symmetric functions are fooled by expander random walks with up to a  $O(\lambda/\sqrt{\rho_{\min}})$  error in total variation distance.

Recently, Golowich and Vadhan [8], significantly strengthened and extended these results using new techniques, and in some cases managed to eliminate the dependence on the bias. In particular, they prove that for the Boolean case, all symmetric functions are fooled by expander random walks with up to  $O(\lambda)$  error in total variation distance, where the constant hidden in the Big-O notation is absolute.

For the non-Boolean case much less is known:

- For *symmetric* functions defined on  $\mathbb{Z}_d^t$ , Golowich and Vadhan prove an  $O\left(\left(\frac{d}{\rho_{\min}}\right)^{O(d)} \cdot \lambda\right)$  total-variation bound where  $\rho_{\min} = \min_a \rho_a$ , and  $\rho_a$  is the density of label  $a$ . Notice that in this bound there is a dependence on  $\rho_{\min}$ . It is an intriguing open problem whether the dependence on the bias is necessary.
- Golowich and Vadhan [8] also show that expander random walks fool width- $w$  *permutation* branching programs up to a  $O(\lambda)$  error in  $\ell_2$  distance, and a  $O(\sqrt{w} \cdot \lambda)$  error in total variation distance, a bound that does not depend on the bias of the labeling. Notice that this bias-independent bound also holds for non-symmetric functions, as long as they are computed by a low-width permutation branching program.

In this work we add another example where the error bound does not depend on the labeling bias. We show for the confinement problem, when the set of maximal density  $\rho(S)$  is also of maximal variance (the variance is  $\sqrt{\rho(S)(1-\rho(S))}$ ), the error bound is  $O(\lambda)$  regardless of the densities. Note that this case is not symmetric. We also improve the lower bound for the symmetric case, as previously discussed.

There are many open problems left.

- First, and foremost, is it possible that all symmetric functions over  $\Sigma^t$  are  $O_{|\Sigma|}(\lambda)$  fooled by random-walks? For  $\Sigma = \{0, 1\}$  [8] gave an affirmative answer, but the general case is left open.
- What other non-symmetric functions are fooled by random-walks without a dependence on the bias? [8] showed all small-width permutation branching programs are such. We added the confinement test functions when all sets have the same variance. What other functions have this property?
- As alluded to by Proposition 7, we think that for many functions the parameter dominating the bias-dependent error is  $1 - \rho_{\max}$  rather than  $\rho_{\min}$ . For example, the bias-dependent bound for *any* confinement test function (Proposition 29) is  $O\left(\frac{\lambda}{1-\rho_{\max}}\right)$  where  $\rho_{\max}$  is the density of the largest set. It would be interesting to examine previous results and see if the error terms can be correspondingly amended.

The paper is organized as follows: In Section 2 we give some preliminaries and background, and introduce our notations. In Section 3 we review Beck's flow framework [3] and extend it. In Section 4 we prove Theorem 3, and prove analogous lower bounds in the general setting of varying sets and densities. In Section 5 we study fooling confinement test functions, and in particular prove Theorem 4. In Section 6, we prove Theorem 6 using our techniques. The proof for Proposition 7 appears in the full version of this paper [14].

## 2 Preliminaries

### Notation

For any positive integer  $d$ , let  $\mathbb{Z}_d$  denote the group of integers modulo  $d$ , and  $[d] = \{1, \dots, d\}$ . We define the  $\ell_1$ -norm of a vector  $x \in \mathbb{F}^n$  as  $\|x\|_1 = \sum_i |x_i|$ , and its  $\ell_2$ -norm as  $\|x\| = \sqrt{\sum_i |x_i|^2}$ . For a field  $\mathbb{F} = \mathbb{R}$  or  $\mathbb{C}$ , let  $\mathbf{1}_n = (1/\sqrt{n}, \dots, 1/\sqrt{n}) \in \mathbb{F}^n$  denote the normalized

all-ones vector. When  $n$  is clear from context we simply write  $\mathbf{1}$ . For a matrix  $M \in \mathbb{F}^{n \times n}$ , the operator norm of  $M$  is given by  $\max_{x \in \mathbb{F}^n \setminus \{0\}} \|Mx\| / \|x\|$ . For  $M \in \mathbb{C}^{n \times n}$ , its conjugate transpose is denoted as  $M^*$ . For two real matrices  $L, M \in \mathbb{R}^{n \times n}$ , the notation  $L \leq_{e.w} M$  stands for entry-wise inequality

A symmetric matrix  $W \in [0, 1]^{n \times n}$  is an undirected random walk matrix on  $n$  vertices if the columns and rows of  $W$  sum to 1, which implies that  $W_{j,i} = W_{i,j}$  represents the transition probability between vertex  $i$  and  $j$ , or vice versa. In this context,  $I_n$  denotes the  $n \times n$  identity matrix, and  $J_n = \mathbf{1}_n \mathbf{1}_n^T$  represents a matrix with all entries being  $1/n$ . When the dimension is clear from the context, we use the notations  $I$  and  $J$  respectively. Notably,  $J_n$  is the random walk matrix for a complete graph on  $n$  vertices with self-loops. For a sequence of matrices  $M_1, \dots, M_t$ , we denote  $\prod_{i=1}^t M_i = M_t \cdot M_{t-1} \cdot \dots \cdot M_1$ .

We often use the decomposition  $\mathbb{F}^n = \mathbf{V}_0 \oplus \mathbf{V}_1$  where  $\mathbf{V}_0 = \text{Span}\{\mathbf{1}\}$  is the subspace of  $\mathbb{F}^n$  spanned of the all ones vector, and  $\mathbf{V}_1 = \mathbf{V}_0^\perp$  is its orthogonal complement. We define  $\Pi_0$  as the projection operator onto  $\mathbf{V}_0$ , noting that  $\Pi_0 = J_n$ , and  $\Pi_1$  as the projection on  $\mathbf{V}_1$ , noting that  $\Pi_1 = I_n - J_n$ . For a vector  $x \in \mathbb{F}^n$  we define  $x^\parallel = \Pi_0 x$  and  $x^\perp = \Pi_1 x$ .

For two probability distributions  $p_1$  and  $p_2$  over a finite sample space  $\Omega$ , their total variation distance is  $d_{TV}(p_1, p_2) = \frac{1}{2} \cdot \sum_{s \in \Omega} |p_1(s) - p_2(s)|$ .

### The Information Theoretic XOR-Lemma

The characters of the group  $\mathbb{Z}_d$  are the maps  $\chi_b(a) = \omega_d^{b \cdot a}$  for  $b = 0, \dots, d - 1$ , where  $\omega_d = e^{\frac{2\pi i}{d}}$ . Let  $\mathbb{C}^{\mathbb{Z}_d}$  denote the vector space of all complex valued function on  $\mathbb{Z}_d$ , equipped with the inner product  $\langle h, g \rangle = \sum_{a \in \mathbb{Z}_d} h(a) \overline{g(a)}$ .

The information theoretic XOR-Lemma [7] relates the total variation distance between two distributions over  $\mathbb{Z}_d$  to the heaviest Fourier coefficient of their difference, also called the maximum bias.

► **Lemma 8** (Based on [7]). *For any two distributions  $p_1, p_2$  over  $\mathbb{Z}_d$ :  $d_{TV}(p_1, p_2) \leq \frac{\sqrt{d}}{2} \cdot \max_{b \in \mathbb{Z}_d} |\langle \chi_b, p_1 - p_2 \rangle|$ .*

The proof, based on [7], appears in the full version of this paper.

### Expanders

For a regular, undirected graph  $G = (V, E)$  on  $n$  vertices, the random walk matrix is the normalized adjacency matrix. The *spectral expansion* is defined as the second largest eigenvalue of the graph's random walk matrix in absolute value, namely  $\lambda(G) = \max_{x, y \perp \mathbf{1}} \frac{|\langle x, Gy \rangle|}{\|x\| \cdot \|y\|} = \max_{x \perp \mathbf{1}} \frac{\|Gx\|}{\|x\|}$ , where the maximum is over all non-zero  $x, y \in \mathbb{R}^n$  which are orthogonal to the all-ones vector, and by abuse of notation  $G$  also denotes the random walk matrix of the graph  $G$ . We say  $G$  is a  $\lambda$ -expander if  $\lambda(G) = \lambda$ . For a  $\lambda$ -expander  $G$ , let  $A = \frac{1}{\lambda}(G - J)$ . Since the all-ones vector is an eigenvector of both  $G$  and  $J$  with eigenvalue 1, it follows that  $A$  is zero on the parallel space  $\text{Span}\{\mathbf{1}\}$ . Additionally,  $\|Ax\| = \|Ax^\perp + Ax^\parallel\| = \frac{1}{\lambda} \cdot \|Gx^\perp\| \leq \|x^\perp\| \leq \|x\|$ . This implies a valuable decomposition  $G = J + \lambda A$  where the symmetric “error matrix”  $A$  is zero on the parallel space, and  $\|A\| \leq 1$ . Another useful decomposition follows by setting  $E = \frac{1}{\lambda}(G - (1 - \lambda) \cdot J)$ . One can easily verify that  $E$  acts like the identity on the parallel space, and that the orthogonal space is  $E$ -invariant. Thus, for every vector  $x$  we have

$$\|Ex\|^2 = \|Ex^\parallel\|^2 + \|Ex^\perp\|^2 \leq \|x^\parallel\|^2 + \frac{1}{\lambda} \|Gx^\perp\|^2 \leq \|x\|^2.$$

This gives rise to the decomposition  $G = (1 - \lambda)J + \lambda E$  where the symmetric “error matrix”  $E$  satisfies  $\|E\| \leq 1$ .

### 3 Flow

Let  $\mathbb{F}$  be either  $\mathbb{C}$  or  $\mathbb{R}$ . We decompose  $\mathbb{F}^n = \mathbf{V}_0 \oplus \mathbf{V}_1$  where  $\mathbf{V}_0$  is the span of the all-ones vector  $\text{Span}\{\mathbf{1}\}$  (the “parallel” space) and  $\mathbf{V}_1 = \mathbf{V}_0^\perp$  its orthogonal complement (the “orthogonal” space). Let  $\Pi_0$  be the projection operator onto  $\mathbf{V}_0$ , and  $\Pi_1$  the projection onto  $\mathbf{V}_1$ .

Throughout this work we study linear operators  $T : \mathbb{F}^n \rightarrow \mathbb{F}^n$  by examining  $\|\Pi_{b_1} T \Pi_{b_2}\|$  for  $b_1, b_2 \in \{0, 1\}$ . Intuitively, this can be understood as the “flow of mass” from  $\mathbf{V}_{b_2}$  to  $\mathbf{V}_{b_1}$  under the linear operator  $T$ . To study the flow of a linear operator, we extend upon the techniques introduced by Gillman, Healy, and Beck, using the notation and claims of Beck [3]. These were used mostly in the context of the expander Chernoff bound [6, 9].

► **Definition 9** (The Flow Matrix). *Let  $T : \mathbb{F}^n \rightarrow \mathbb{F}^n$  be any linear operator. Then the flow matrix of  $T$ , denoted  $\tilde{T}$ , is the  $2 \times 2$  non-negative matrix defined by*

$$\tilde{T} = \begin{pmatrix} \|\Pi_0 T \Pi_0\| & \|\Pi_0 T \Pi_1\| \\ \|\Pi_1 T \Pi_0\| & \|\Pi_1 T \Pi_1\| \end{pmatrix}$$

► **Example 10.** Let  $G$  be the random walk operator of a  $\lambda$ -expander graph. Then

$$\tilde{G} \leq_{e.w} \begin{pmatrix} 1 & 0 \\ 0 & \lambda \end{pmatrix}$$

where  $\leq_{e.w}$  stands for entry-wise inequality.

To see this, apply the decomposition  $G = J + \lambda A$  where  $\|A\| \leq 1$  and  $A$  is zero on the parallel space. That is,  $A\Pi_0 = \Pi_0 A = 0$ . We then have

- (i)  $\|\Pi_0 G \Pi_0\| = \|\Pi_0 J \Pi_0\| = \|J\| = 1$ ,
- (ii)  $\|\Pi_0 G \Pi_1\| = \|\Pi_0 (J + \lambda A) \Pi_1\| = \|\Pi_0 J \Pi_1 + \lambda \Pi_0 A \Pi_1\| = 0$ ,
- (iii) By symmetry  $\|\Pi_1 G \Pi_0\| = 0$ .
- (iv) Finally,  $\|\Pi_1 G \Pi_1\| = \lambda \|\Pi_1 A \Pi_1\| \leq \lambda$ .

By submultiplicativity and subadditivity of the operator norm, we have the following submultiplicativity property of the flow operator:

► **Claim 11** ([3]). For every linear operators  $L, M : \mathbb{F}^n \rightarrow \mathbb{F}^n$ , we have  $\widetilde{L \cdot M} \leq_{e.w} \tilde{L} \cdot \tilde{M}$ .

Proof. Let  $i, j \in \{0, 1\}$ . Recall that  $\Pi_0 = J$  and  $\Pi_1 = I - J$ , and thus  $\Pi_0 + \Pi_1 = I$ . We have

$$\begin{aligned} \widetilde{L \cdot M}[i, j] &= \|\Pi_i L M \Pi_j\| = \|\Pi_i L (\Pi_0 + \Pi_1) M \Pi_j\| \leq \|\Pi_i L \Pi_0 M \Pi_j\| + \|\Pi_i L \Pi_1 M \Pi_j\| \\ &\leq \|\Pi_i L \Pi_0\| \cdot \|\Pi_0 M \Pi_j\| + \|\Pi_i L \Pi_1\| \cdot \|\Pi_1 M \Pi_j\| \\ &= \tilde{L}[i, 0] \cdot \tilde{M}[0, j] + \tilde{L}[i, 1] \cdot \tilde{M}[1, j] = \tilde{L} \cdot \tilde{M}[i, j]. \end{aligned} \quad \triangleleft$$

Typically, the primary technical tool utilized for analyzing flow matrices consists of the following bound, which generally hold for non-negative  $2 \times 2$  matrices.

► **Lemma 12** ([3]). *If  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \geq_{e.w} 0$  with  $a \geq 1$  and  $d < 1$ , then*

$$A^t[0, 0] \leq a \cdot \left( a + \frac{bc}{1-d} \right)^{t-1}$$

**Proof.** By induction on  $t$ . The base case  $t = 1$  is clear. Assume for  $1, \dots, t-1$  and let us prove for  $t$ . We have the following recurrence relation

$$A^t[0, 0] = A^{t-1}[0, 0] \cdot A[0, 0] + \sum_{j=0}^{t-2} A^j[0, 0] \cdot A[0, 1] \cdot A[1, 1]^{t-2-j} \cdot A[1, 0]$$

where  $j$  goes over the last time the path was at vertex 0 before taking the final step. As  $A[i, j] \geq 0$  and  $A[0, 0] \geq 1$ , we see that  $A^{k_2}[0, 0] \geq A^{k_1}[0, 0]$  for all  $k_2 \geq k_1$ . Hence,

$$\begin{aligned} A^t[0, 0] &= A^{t-1}[0, 0] \cdot a + \sum_{j=0}^{t-2} A^j[0, 0] \cdot bc \cdot d^{t-2-j} \\ &\leq A^{t-1}[0, 0] \left( a + bc \sum_{j=0}^{\infty} d^j \right) \leq A^{t-1}[0, 0] \left( a + \frac{bc}{1-d} \right) \end{aligned}$$

The proof is complete by applying the induction hypothesis.  $\blacktriangleleft$

A simple way to generalize this lemma to the case where  $A[0, 0] > A[1, 1]$  but not necessarily  $A[0, 0] > 1$  is as follows.

► **Lemma 13.** *If  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \geq_{e.w} 0$  with  $a > d$  then  $A^t[0, 0] \leq a \cdot \left( a + \frac{bc}{a-d} \right)^{t-1}$ .*

**Proof.** Write  $A = a \cdot \begin{pmatrix} 1 & \frac{b}{a} \\ \frac{c}{a} & \frac{d}{a} \end{pmatrix}$ . Then, by the previous lemma

$$A^t[0, 0] \leq a^t \cdot \left( 1 + \frac{\frac{b}{a} \cdot \frac{c}{a}}{1 - \frac{d}{a}} \right)^{t-1} = a \cdot \left( a + \frac{bc}{a-d} \right)^{t-1}. \quad \blacktriangleleft$$

► **Remark 14.** Note that the lemma above is not tight when  $a$  is small. Indeed,  $A^t[0, 0]$  decreases with  $a$ , while the bound of Lemma 13 blows up when  $a$  approaches  $d$ . We do not try to optimize the bound for  $d$  close to  $a$ . Also, it would be nice to have a generalization of this lemma for the case of possibly different  $A_1, \dots, A_t$ .

► **Lemma 15.** *If  $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \geq_{e.w} 0$  with  $a > d$ . Then for all  $t \geq 2$ ,*

$$A^t[0, 0] - (A[0, 0])^t \leq \frac{abc}{a-d} \cdot \sum_{k=0}^{t-2} a^k \left( a + \frac{bc}{a-d} \right)^{t-k-2}$$

**Proof.** For every integer  $k \geq 1$   $x^k - y^k = (x-y) \cdot \sum_{i=0}^{k-1} x^i y^{k-i-1}$ . Using this and Lemma 13, we see that

$$\begin{aligned} A^t[0, 0] - (A[0, 0])^t &\leq a \cdot \left( a + \frac{bc}{a-d} \right)^{t-1} - a^t = a \left( \left( a + \frac{bc}{a-d} \right)^{t-1} - a^{t-1} \right) \\ &= a \cdot \frac{bc}{a-d} \cdot \sum_{k=0}^{t-2} a^k \left( a + \frac{bc}{a-d} \right)^{t-k-2} \quad \blacktriangleleft \end{aligned}$$

► **Lemma 16.** *For an integer  $t \geq 1$ , Let  $M_1, \dots, M_t$  be a sequence of  $n \times n$  matrices. Then*

$$\left| \mathbf{1}^T \left( \prod_{i=1}^t M_i \right) \mathbf{1} - \mathbf{1}^T \left( \prod_{i=1}^t \Pi_0 M_i \right) \mathbf{1} \right| \leq \left( \prod_{i=1}^t \widetilde{M}_i \right) [0, 0] - \left( \prod_{i=1}^t \widetilde{M}_i [0, 0] \right).$$

### 31:10 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

**Proof.** Writing  $M_i = \Pi_0 M_i + \Pi_1 M_i$  we have

$$\mathbf{1}^T \left( \prod_{i=1}^t M_i \right) \mathbf{1} = \sum_{b \in \{0,1\}^t} \mathbf{1}^T \prod_{i=1}^t (\Pi_{b_i} M_i) \mathbf{1} = \sum_{b \in \{0,1\}^{t-1}} \mathbf{1}^T \Pi_0 M_t \left( \prod_{i=1}^{t-1} (\Pi_{b_i} M_i) \right) \cdot \mathbf{1}$$

Since  $\mathbf{1}^T \Pi_1 = 0$ . To complete the proof let  $LHS = \left| \mathbf{1}^T \prod_{i=1}^t M_i \mathbf{1} - \mathbf{1}^T \left( \prod_{i=1}^t (\Pi_0 M_i) \right) \mathbf{1} \right|$ . Then,

$$\begin{aligned} LHS &= \left| \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \mathbf{1}^T \Pi_0 M_t \left( \prod_{i=1}^{t-1} (\Pi_{b_i} M_i) \right) \cdot \mathbf{1} \right| \\ &\leq \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \left| \mathbf{1}^T \Pi_0 M_t \left( \prod_{i=1}^{t-1} (\Pi_{b_i} M_i) \right) \cdot \Pi_0 \mathbf{1} \right| \leq \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \left\| \Pi_0 M_t \left( \prod_{i=1}^{t-1} (\Pi_{b_i} M_i) \right) \cdot \Pi_0 \right\| \\ &= \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \left\| \Pi_0 M_t \Pi_{b_{t-1}} \left( \prod_{i=2}^{t-1} (\Pi_{b_i} M_i \Pi_{b_{i-1}}) \right) \cdot \Pi_{b_1} M_1 \Pi_0 \right\| \\ &\leq \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \left\| \Pi_0 M_t \Pi_{b_{t-1}} \right\| \cdot \prod_{i=2}^{t-1} \left\| \Pi_{b_i} M_i \Pi_{b_{i-1}} \right\| \left\| \Pi_{b_1} M_1 \Pi_0 \right\| \\ &= \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} \widetilde{M}_t[0, b_{t-1}] \left( \prod_{i=2}^{t-1} \widetilde{M}_i[b_i, b_{i-1}] \right) \widetilde{M}_1[b_1, 0] = \left( \prod_{i=1}^t \widetilde{M}_i \right) [0, 0] - \prod_{i=1}^t \widetilde{M}_i[0, 0]. \quad \blacktriangleleft \end{aligned}$$

► **Lemma 17.** Let  $A_1, \dots, A_t$  be a sequence of non-negative  $2 \times 2$  matrices such that for all  $i$ ,  $A_i \leq_{e.w} A$  for some  $2 \times 2$  matrix  $A$ . Then

$$\left( \prod_{i=1}^t A_i \right) [0, 0] - \prod_{i=1}^t (A_i[0, 0]) \leq A^t[0, 0] - (A[0, 0])^t.$$

**Proof.** We have

$$\begin{aligned} \left( \prod_{i=1}^t A_i \right) [0, 0] - \prod_{i=1}^t (A_i[0, 0]) &= \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^t}} A_t[0, b_{t-1}] \left( \prod_{i=2}^{t-1} A_i[b_i, b_{i-1}] \right) A_1[b_1, 0] \\ &\leq \sum_{\substack{b \in \{0,1\}^{t-1} \\ b \neq 0^{t-1}}} A[0, b_{t-1}] \left( \prod_{i=2}^{t-1} A[b_{i+1}, b_i] \right) A[b_1, 0] = A^t[0, 0] - (A[0, 0])^t. \quad \blacktriangleleft \end{aligned}$$

We now proceed to establish techniques for proving flow lower bounds. While these concepts were introduced specifically for the confinement problem with the same set density in [2], we extend them to general linear operators and use the flow matrix notation.

► **Lemma 18** (Flow Progress). *For linear operators  $T_1, \dots, T_t$ ,*

$$\widetilde{\prod_{i=1}^t T_i[0, 0]} \geq \widetilde{T_t[0, 0]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[0, 0]} - \widetilde{T_t[0, 1]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[1, 0]} \quad (2)$$

$$\widetilde{\prod_{i=1}^t T_i[1, 0]} \leq \widetilde{T_t[1, 0]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[0, 0]} + \widetilde{T_t[1, 1]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[1, 0]} \quad (3)$$

**Proof.** We have

$$\begin{aligned} \widetilde{\prod_{i=1}^t T_i[0, 0]} &= \left\| \Pi_0 \prod_{i=1}^t T_i \Pi_0 \right\| = \left\| \Pi_0 (T_t \Pi_0 + T_t \Pi_1) \prod_{i=1}^{t-1} T_i \Pi_0 \right\| \\ &\geq \left\| \Pi_0 T_t \Pi_0 \right\| \cdot \left\| \prod_{i=1}^{t-1} T_i \Pi_0 \right\| - \left\| \Pi_0 T_t \Pi_1 \right\| \cdot \left\| \prod_{i=1}^{t-1} T_i \Pi_0 \right\| \\ &= \widetilde{T_t[0, 0]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[0, 0]} - \widetilde{T_t[0, 1]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[1, 0]} \end{aligned}$$

and

$$\begin{aligned} \widetilde{\prod_{i=1}^t T_i[1, 0]} &= \left\| \Pi_1 \prod_{i=1}^t T_i \Pi_0 \right\| = \left\| \Pi_1 (T_t \Pi_0 + T_t \Pi_1) \prod_{i=1}^{t-1} T_i \Pi_0 \right\| \\ &\leq \left\| \Pi_1 T_t \Pi_0 \right\| \cdot \left\| \prod_{i=1}^{t-1} T_i \Pi_0 \right\| + \left\| \Pi_1 T_t \Pi_1 \right\| \cdot \left\| \prod_{i=1}^{t-1} T_i \Pi_0 \right\| \\ &= \widetilde{T_t[1, 0]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[0, 0]} + \widetilde{T_t[1, 1]} \cdot \widetilde{\prod_{i=1}^{t-1} T_i[1, 0]} \quad \blacktriangleleft \end{aligned}$$

► **Definition 19** (Flow sequence). *For a sequence of linear operators  $T_1, \dots, T_t$ , the flow sequence is defined recursively such that  $c_1 = \frac{\widetilde{T_1[0, 0]}}{\widetilde{T_1[1, 0]}}$  and for  $k \geq 1$*

$$c_{k+1} = \frac{\widetilde{T_{k+1}[0, 0]} \cdot c_k - \widetilde{T_{k+1}[0, 1]}}{\widetilde{T_{k+1}[1, 0]} \cdot c_k + \widetilde{T_{k+1}[1, 1]}}$$

The constants  $c_i$  emerge from recursively dividing Equation 2 of Lemma 18 by Equation 3, as demonstrated by the following lemmas. Therefore, from an intuitive perspective, the constants  $c_i$  in the definition above can be thought of as a lower bound on the ratio between the mass preserved inside the parallel space after the  $i$ -th step and the mass lost to its orthogonal complement.

We remark that the smaller  $\widetilde{T_i[0, 1]}$ ,  $\widetilde{T_i[1, 1]}$  are taken relative to  $\widetilde{T_i[0, 0]}$  and  $\widetilde{T_i[1, 0]}$ , the larger sequence elements will become. In all of our use cases, each operator  $T_i$  includes a step on a  $\lambda$ -expander graph  $G$ . Thus, as we shall later see, we can make  $\widetilde{T_i[0, 1]}$  and  $\widetilde{T_i[1, 1]}$  smaller by taking the expansion parameter  $\lambda$  smaller, and hence the sequence elements larger. Specifically, in all instances considered in this work, the constants  $c_i$  are strictly positive. Therefore, for the remainder of this section, we proceed with the assumption that the provided linear operators  $T_1, \dots, T_t$  are such that their corresponding flow sequence elements are positive.

## 31:12 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

► **Lemma 20.** *Let  $T_1, \dots, T_t$  be linear operators with a positive flow sequence. Then, for all  $k = 1, \dots, t$  it holds that  $\widetilde{\prod_{i=1}^k T_i[0, 0]} \geq c_k \cdot \widetilde{\prod_{i=1}^k T_i[1, 0]}$ .*

**Proof.** By induction on  $k$ . For  $k = 1$  the claim holds by definition. For the induction step, assume that  $\widetilde{\prod_{i=1}^k T_i[0, 0]} \geq c_k \cdot \widetilde{\prod_{i=1}^k T_i[1, 0]}$ . Plugging the induction hypothesis into Equation 2 see that

$$\begin{aligned} \widetilde{\prod_{i=1}^{k+1} T_i[0, 0]} &\geq \widetilde{T_{k+1}[0, 0]} \cdot \widetilde{\prod_{i=1}^k T_i[0, 0]} - \widetilde{T_{k+1}[0, 1]} \cdot \widetilde{\prod_{i=1}^k T_i[1, 0]} \\ &\geq \left( \widetilde{T_{k+1}[0, 0]} - \frac{\widetilde{T_{k+1}[0, 1]}}{c_k} \right) \widetilde{\prod_{i=1}^{k-1} T_i[0, 0]}. \end{aligned}$$

Similarly, plugging the induction hypothesis into Equation 3,

$$\begin{aligned} \widetilde{\prod_{i=1}^{k+1} T_i[1, 0]} &\leq \widetilde{T_{k+1}[1, 0]} \cdot \widetilde{\prod_{i=1}^k T_i[0, 0]} + \widetilde{T_{k+1}[1, 1]} \cdot \widetilde{\prod_{i=1}^k T_i[1, 0]} \\ &\leq \left( \widetilde{T_{k+1}[1, 0]} + \frac{\widetilde{T_{k+1}[1, 1]}}{c_k} \right) \widetilde{\prod_{i=1}^k T_i[0, 0]}. \end{aligned}$$

Combining these we obtain

$$\begin{aligned} \widetilde{\prod_{i=1}^{k+1} T_i[0, 0]} &\geq \frac{\left( \widetilde{T_{k+1}[0, 0]} - \frac{\widetilde{T_{k+1}[0, 1]}}{c_k} \right) \widetilde{\prod_{i=1}^k T_i[0, 0]}}{\left( \widetilde{T_{k+1}[1, 0]} + \frac{\widetilde{T_{k+1}[1, 1]}}{c_k} \right)} \widetilde{\prod_{i=1}^k T_i[0, 0]} \\ &= \left( \frac{\widetilde{T_{k+1}[0, 0]} \cdot c_k - \widetilde{T_{k+1}[0, 1]}}{\widetilde{T_{k+1}[1, 0]} \cdot c_k + \widetilde{T_{k+1}[1, 1]}} \right) \widetilde{\prod_{i=1}^k T_i[0, 0]} = c_{k+1} \cdot \widetilde{\prod_{i=1}^k T_i[0, 0]} \quad \blacktriangleleft \end{aligned}$$

Hence we have the following corollary

► **Corollary 21.** *For all  $k = 1, \dots, t$  we have*

$$\widetilde{\prod_{i=1}^k T_i[0, 0]} \geq \widetilde{T_1[0, 0]} \cdot \prod_{i=2}^k \left( \widetilde{T_i[0, 0]} - \frac{\widetilde{T_i[0, 1]}}{c_{i-1}} \right)$$

**Proof.** By induction on  $k$ . For  $k = 1$  the product on the right hand side is empty and the equality trivially holds. For the induction step, Using Equation 2, the previous claim, and the induction hypothesis,

$$\begin{aligned} \widetilde{\prod_{i=1}^{k+1} T_i[0, 0]} &\geq \widetilde{T_{k+1}[0, 0]} \cdot \widetilde{\prod_{i=1}^k T_i[0, 0]} - \widetilde{T_{k+1}[0, 1]} \cdot \widetilde{\prod_{i=1}^k T_i[1, 0]} \\ &\geq \left( \widetilde{T_{k+1}[0, 0]} - \frac{\widetilde{T_{k+1}[0, 1]}}{c_k} \right) \widetilde{\prod_{i=1}^k T_i[0, 0]} \geq \widetilde{T_1[0, 0]} \cdot \prod_{i=2}^{k+1} \left( \widetilde{T_i[0, 0]} - \frac{\widetilde{T_i[0, 1]}}{c_{i-1}} \right). \quad \blacktriangleleft \end{aligned}$$



## 4 Expander Hitting Property Revised

We use the following notations. For a set  $S_i \subseteq [n]$  we define its density as  $\rho_i = |S_i|/n$  and its variance as  $\sigma_i = \sqrt{\rho_i(1-\rho_i)}$ . We let  $P_i$  be the projection matrix on the set  $S_i$ . That is,  $P_i$  is the diagonal matrix satisfying  $P_i[v, v] = 1$  if  $v \in S_i$  and 0 otherwise.  $G$  is the random walk operator of the graph  $G$ .

### 4.1 Confinement Probability Upper-bounds

We begin with the hitting property for sets with possibly different densities. In [10] the authors give the bound  $\prod_{j=1}^{t-1} (\sqrt{\rho_j \rho_{j+1}} + \lambda)$ , which corresponds to  $\|P_t G \dots G P_1\|$  rather than  $\mathbf{1}^T P_t G \dots G P_1 \mathbf{1}$ . However, we observe that this loss is not necessary.

► **Proposition 22** (Expander Hitting Property). *Let  $G = (V, E)$  be a  $\lambda$ -expander. Then, for every sequence of subsets  $S_1, \dots, S_t \subseteq V$  such that  $S_i$  is of density  $\rho_i$ ,*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] \leq \sqrt{\rho_1 \rho_t} \cdot \prod_{i=1}^{t-1} ((1-\lambda)\sqrt{\rho_i \rho_{i+1}} + \lambda).$$

**Proof.** First note that for all  $i$ ,  $\|P_i J P_{i+1}\| = \sqrt{\rho_i \rho_{i+1}}$ . Indeed,

$$\|P_i J P_{i+1}\| = \|P_i \mathbf{1} (P_{i+1} \mathbf{1})^T\| = \|P_i \mathbf{1}\| \cdot \|P_{i+1} \mathbf{1}\| = \sqrt{\rho_i \rho_{i+1}}$$

Decomposing  $G = (1-\lambda)J + \lambda E$  with  $\|E\| \leq 1$ , we find that

$$\begin{aligned} \|P_i G P_{i+1}\| &= \|(1-\lambda) \cdot P_i J P_{i+1} + \lambda \cdot P_i E P_{i+1}\| \\ &\leq (1-\lambda) \cdot \|P_i J P_{i+1}\| + \lambda \leq (1-\lambda)\sqrt{\rho_i \rho_{i+1}} + \lambda. \end{aligned}$$

Let  $u = (1/n, \dots, 1/n) \in \mathbb{R}^n$  be the uniform vector. Expressing the probability linear-algebraically we obtain

$$\begin{aligned} \Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] &= \mathbf{1}^T P_t \left( \prod_{i=1}^{t-1} G P_i \right) \mathbf{1} = \mathbf{1}^T P_t \prod_{i=1}^{t-1} (P_{i+1} G P_i) P_1 \mathbf{1} \\ &= \left\| P_t \prod_{i=1}^{t-1} (P_{i+1} G P_i) P_1 u \right\|_1 \leq \sqrt{\rho_t \cdot n} \cdot \left\| \prod_{i=1}^{t-1} (P_{i+1} G P_i) P_1 u \right\| \\ &\leq \sqrt{\rho_t \cdot n} \cdot \left\| \prod_{i=1}^{t-1} (P_{i+1} G P_i) \right\| \cdot \|P_1 u\| = \sqrt{\rho_t \cdot n} \cdot \left\| \prod_{i=1}^{t-1} (P_{i+1} G P_i) \right\|_2 \cdot \sqrt{\frac{\rho_1}{n}} \\ &\leq \sqrt{\rho_t \rho_1} \cdot \prod_{i=1}^{t-1} \|(P_{i+1} G P_i)\| \leq \sqrt{\rho_1 \rho_t} \cdot \prod_{i=1}^{t-1} ((1-\lambda)\sqrt{\rho_i \rho_{i+1}} + \lambda), \end{aligned}$$

where we use  $P_i^2 = P_i$ , and the first inequality is Cauchy-Schwartz, noting that after multiplying by  $P_t$ , the resulting vector has at most  $\rho_t \cdot n$  non-zero entries. ◀

### 4.2 Confinement Probability Lower-bounds

As explained in the introduction, previous lower bounds do not give an  $O(\lambda)$  bound on the error term comparing with the independent sampling case. In this section, we give a tighter lower bound that, in particular, is  $O(\lambda)$ -close to the probability of the same confinement event but with independently chosen samples.

## 31:14 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

Expressing the probability linear-algebraically we find that

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] = \mathbf{1}^T \prod_{i=2}^t (P_i G) P_1 \mathbf{1} = \left\| \Pi_0 \prod_{i=1}^t (P_i G) \Pi_0 \right\| = \widetilde{\prod_{i=1}^t (P_i G)[0, 0]}.$$

Therefore, we see that this quantity is applicable to bounds via the lower-bound part of the flow framework. Consider the sequence of linear operators  $P_1 G, \dots, P_t G$  with corresponding flow sequence  $c_1, \dots, c_t$ . It follows from Corollary 21 that

$$\widetilde{\prod_{i=1}^t (P_i G)[0, 0]} \geq \widetilde{P_1 G}[0, 0] \cdot \prod_{i=2}^{k+1} \left( \widetilde{P_i G}[0, 0] - \frac{\widetilde{P_i G}[0, 1]}{c_{i-1}} \right)$$

Hence, our next objective is to bound the entries of  $\widetilde{P_i G}$ , and find lower bounds on the constants  $c_1, \dots, c_t$ .

► **Lemma 23.** *For all  $i = 1, \dots, t$  we have  $\widetilde{P_i G} \leq_{e.w} \begin{pmatrix} \rho_i & \lambda \sigma_i \\ \sigma_i & \lambda \end{pmatrix}$  where the first column holds with equality.*

**Proof.** First, observe that

$$\widetilde{P_i G}[0, 0] = \|\Pi_0 P_i G \Pi_0\| = \|\mathbf{1} \mathbf{1}^T P_i G \mathbf{1} \mathbf{1}^T\| = \|\mathbf{1} \mathbf{1}^T P_i \mathbf{1} \mathbf{1}^T\| = |\mathbf{1}^T P_i \mathbf{1}| = \rho_i$$

Following the discussion about the norm of rank-one matrices, we see that for  $b \in \{0, 1\}$ ,

$$\|\Pi_b P_i G \Pi_b\| = \|\Pi_b P_i G \mathbf{1} \mathbf{1}^T\| = \|\Pi_b P_i \mathbf{1}\| \cdot \|\mathbf{1}\| = \|\Pi_b P_i \mathbf{1}\|.$$

Using this, we find that

$$\widetilde{P_i G}[0, 0]^2 + \widetilde{P_i G}[1, 0]^2 = \|\Pi_0 P_i \mathbf{1}\|^2 + \|\Pi_1 P_i \mathbf{1}\|^2 = \|P_i \mathbf{1}\|^2 = \rho_i$$

hence  $\widetilde{P_i G}[1, 0] = \sqrt{\rho_i(1 - \rho_i)} = \sigma_i$ .

Now, let us write  $G = J + \lambda A$  where  $\|A\| \leq 1$  and  $A$  is zero on the parallel space. Then

$$\begin{aligned} \widetilde{P_i G}[0, 1] &= \|\Pi_0 P_i G \Pi_1\| = \|\Pi_0 P_i (J + \lambda A) \Pi_1\| = \lambda \|\Pi_0 P_i A \Pi_1\| = \lambda \|\Pi_0 P_i \Pi_1 A \Pi_1\| \\ &\leq \lambda \|\Pi_0 P_i \Pi_1\| = \lambda \sigma_i \end{aligned}$$

where we have used that  $\Pi_1 A = A$  in the last equality. In the inequality we observe that  $\widetilde{P_i G}[1, 0] = \|\Pi_1 P_i G \Pi_0\| = \|\Pi_1 P_i \Pi_0\| = \|\Pi_0 P_i \Pi_1\|$ . Hence we substitute  $\|\Pi_0 P_i \Pi_1\| = \sigma_i$ .

For the last entry we have  $\widetilde{P_i G}[1, 1] = \|\Pi_1 P_i G \Pi_1\| = \lambda \|\Pi_1 P_i A \Pi_1\| \leq \lambda$ . ◀

By definition of flow sequence (Definition 19) and the previous lemma, we obtain:

► **Corollary 24** (Flow sequence lower-bound). *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets such that  $S_i$  is of density  $\rho_i$  and variance  $\sigma_i$ . Let  $c_1, \dots, c_t$  be the flow sequence of the linear operators  $P_1 G, \dots, P_t G$ . Then  $c_1 = \frac{\rho_1}{\sigma_1}$  and:  $c_{i+1} \geq \frac{c_i \rho_{i+1} - \lambda \sigma_{i+1}}{c_i \sigma_{i+1} + \lambda}$ .*

► **Corollary 25.** *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets such that  $S_i$  is of density  $\rho_i$ . Let  $c_1, \dots, c_t$  be the flow sequence of the linear operators  $P_1 G, \dots, P_t G$ . Suppose that  $\lambda$  is sufficiently small so that  $c_i > 0$  for all  $i$ . Then,*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] \geq \rho_1 \cdot \prod_{i=2}^t \left( \rho_i - \frac{\sigma_i}{c_{i-1}} \lambda \right).$$

Next, we demonstrate how distinct conditions imposed on  $\lambda$  lead to varying bounds on the flow sequence, consequently leading to corresponding confinement probability lower bounds.

► **Lemma 26.** *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets each of density  $\rho_i$ . If for all  $i$ ,  $\lambda < \frac{1}{6} \cdot \sigma_i \sigma_{i+1} \cdot \frac{1+\rho_{i+1}}{1-\rho_{i+1}}$ , then*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] \geq \rho_1 \cdot \prod_{i=2}^t \left( \rho_i - 2 \cdot \frac{\sigma_i}{\sigma_{i-1}} \lambda \right).$$

**Proof.** By Corollary 25, it suffices to prove that under our assumption on  $\lambda$ , we have  $c_i \geq \sigma_i/2$  for all  $i$ .

For  $i = 1$ , we clearly have  $c_1 = \frac{\rho_1}{\sigma_1} = \frac{\sigma_1}{1-\rho_1} > \sigma_1$ . Now, assume that  $c_i \geq \sigma_i/2$ . Using Corollary 24, we find that

$$\begin{aligned} c_{i+1} - \frac{\sigma_{i+1}}{2} &\geq \frac{c_i \rho_{i+1} - \lambda \sigma_{i+1}}{c_i \sigma_{i+1} + \lambda} - \frac{\sigma_{i+1}}{2} = \frac{c_i (2\rho_{i+1} - \sigma_{i+1}^2) - 3\lambda \sigma_{i+1}}{2(c_i \sigma_{i+1} + \lambda)} \\ &= \frac{c_i \rho_{i+1} (1 + \rho_{i+1}) - 3\lambda \sigma_{i+1}}{2(c_i \sigma_{i+1} + \lambda)} \end{aligned}$$

Therefore it suffices to show  $3\lambda \sigma_{i+1} \leq c_i \rho_{i+1} (1 + \rho_{i+1})$ . Indeed, using our assumption on  $\lambda$  and the induction hypothesis,

$$\lambda < \frac{1}{6} \cdot \sigma_i \sigma_{i+1} \cdot \frac{1 + \rho_{i+1}}{1 - \rho_{i+1}} \leq \frac{c_i}{3} \cdot \frac{\sigma_{i+1}}{1 - \rho_{i+1}} (1 + \rho_{i+1}) = \frac{c_i}{3} \cdot \frac{\rho_{i+1}}{\sigma_{i+1}} (1 + \rho_{i+1}). \quad \blacktriangleleft$$

The first part of Theorem 2 follows as a special case of the lemma above, in which all sets have the same density. Indeed, in this case, our assumption on  $\lambda$  becomes  $\lambda < \frac{1}{6} \cdot \sigma^2 \cdot \frac{1+\rho}{1-\rho} = \frac{1}{6} \rho (1+\rho)$ .

► **Lemma 27.** *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets each of density  $\rho_i$ . If for all  $i$ ,  $\lambda < \frac{1}{2} \cdot \frac{\sigma_i}{\sigma_{i+1}} \cdot \rho_{i+1}^2$ , then*

$$\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] \geq \rho_1 \cdot \prod_{i=2}^t \left( \rho_i - \frac{\sigma_i}{\sigma_{i-1}} \lambda \right).$$

**Proof.** By Corollary 25 it suffices to prove that under our assumption on  $\lambda$ , we have  $c_i \geq \sigma_i$  for all  $i$ . The proof is by induction on  $i$ . For  $i = 1$  we clearly have  $c_1 = \sqrt{\frac{\rho_1}{1-\rho_1}} = \frac{\sigma_1}{1-\rho_1} > \sigma_1$ . Assume that  $c_i \geq \sigma_i$ . By Corollary 24, we have

$$\begin{aligned} \frac{c_{i+1}}{\sigma_{i+1}} &\geq \frac{c_i \rho_{i+1} - \lambda \sigma_{i+1}}{c_i \sigma_{i+1}^2 + \lambda \sigma_{i+1}} = \frac{c_i \rho_{i+1} + c_i \sigma_{i+1}^2 + \lambda \sigma_{i+1} - c_i \sigma_{i+1}^2 - 2\lambda \sigma_{i+1}}{c_i \sigma_{i+1}^2 + \lambda \sigma_{i+1}} \\ &= 1 + \frac{c_i (\rho_{i+1} - \sigma_{i+1}^2) - 2\lambda \sigma_{i+1}}{c_i \sigma_{i+1}^2 + \lambda \sigma_{i+1}} = 1 + \frac{c_i \rho_{i+1}^2 - 2\lambda \sigma_{i+1}}{c_i \sigma_{i+1}^2 + \lambda \sigma_{i+1}} \end{aligned}$$

Therefore it suffices to show  $2\lambda \sigma_{i+1} \leq c_i \rho_{i+1}^2$ . Indeed, using our assumption on  $\lambda$  and the induction hypothesis,  $\lambda < \frac{1}{2} \cdot \frac{\sigma_i}{\sigma_{i+1}} \cdot \rho_{i+1}^2 \leq \frac{c_i \rho_{i+1}^2}{2\sigma_{i+1}}$ .  $\blacktriangleleft$

The second part of Theorem 2 follows as a special case of the lemma above, in which all sets have the same density. In that case our assumption on  $\lambda$  becomes  $\lambda < \frac{\rho^2}{2}$ .

The following lemma refines the bound given in [2] and also allows for arbitrary densities with decreasing variances.

## 31:16 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

► **Lemma 28.** *Let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t \subseteq V$  be a sequence of subsets, each of density  $\rho_i$  and variance  $\sigma_i$ . Suppose that  $\sigma_1 \geq \dots \geq \sigma_t$ . If  $\lambda \leq \frac{\sigma_i}{\sigma_{i-1}} \cdot \frac{\rho_{i-1}\rho_i}{4}$  for all  $i$ , then  $\Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] \geq \rho_1 \cdot \prod_{i=2}^t (\rho_i - \lambda(1 - \rho_{i-1}^2))$ .*

**Proof.** Using our assumption that  $\sigma_i \geq \sigma_{i+1}$  for all  $i = 1, \dots, t$ , it suffices to prove that  $\sigma_i \leq (1 - \rho_i^2)c_i$  for all  $i$ . Indeed, in that case, by Corollary 25 we obtain

$$\begin{aligned} \Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i \ v_i \in S_i] &\geq \rho_1 \cdot \prod_{i=2}^t \left( \rho_i - \frac{\sigma_i}{c_{i-1}} \lambda \right) \geq \rho_1 \cdot \prod_{i=2}^t \left( \rho_i - \frac{\sigma_{i-1}}{c_{i-1}} \lambda \right) \\ &\geq \rho_1 \cdot \prod_{i=2}^t (\rho_i - (1 - \rho_{i-1}^2) \cdot \lambda) \end{aligned}$$

Now, we prove by induction on  $i$  that  $\sigma_i \leq (1 - \rho_i^2)c_i$ . For  $i = 1$  we clearly have

$$(1 - \rho_1^2)c_1 = (1 - \rho_1^2) \sqrt{\frac{\rho_1}{1 - \rho_1}} = (1 + \rho_1) \sqrt{\rho_1(1 - \rho_1)} > \sigma_1.$$

Assume that  $\sigma_{i-1} \leq (1 - \rho_{i-1}^2)c_{i-1}$ . Then, by Corollary 24

$$\begin{aligned} \frac{1 - \rho_i^2}{\sigma_i} \cdot c_i &\geq \frac{1 - \rho_i^2}{\sigma_i} \cdot \frac{c_{i-1}\rho_i - \lambda\sigma_i}{c_{i-1}\sigma_i + \lambda} = \frac{c_{i-1}\sigma_i(1 + \rho_i) - \lambda(1 - \rho_i^2)}{c_{i-1}\sigma_i + \lambda} \\ &= \frac{c_{i-1}\sigma_i + \lambda + c_{i-1}\sigma_i\rho_i - \lambda(2 - \rho_i^2)}{c_{i-1}\sigma_i + \lambda} = 1 + \frac{c_{i-1}\sigma_i\rho_i - \lambda(2 - \rho_i^2)}{c_{i-1}\sigma_i + \lambda} \end{aligned}$$

where the second equality uses the identity  $\rho(1 - \rho^2)/\sigma = \sigma(1 + \rho)$ . Thus, it remains to prove that  $c_{i-1}\sigma_i\rho_i \geq \lambda(2 - \rho_i^2)$ . Indeed, on the one hand, by our induction hypothesis

$$c_{i-1}\sigma_i\rho_i \geq \frac{\sigma_{i-1}\sigma_i\rho_i}{(1 - \rho_{i-1}^2)} = \frac{\sigma_i}{\sigma_{i-1}} \cdot \frac{\rho_{i-1}\rho_i}{1 + \rho_{i-1}}$$

using the identity  $\rho/\sigma = \sigma/(1 - \rho)$ .

On the other hand, our assumption on  $\lambda$  implies that

$$\lambda \leq \frac{\sigma_i}{\sigma_{i-1}} \cdot \frac{\rho_{i-1}\rho_i}{4} \leq \frac{\sigma_i}{\sigma_{i-1}} \cdot \frac{\rho_{i-1}\rho_i}{(1 + \rho_{i-1})(2 - \rho_i^2)}$$

and the proof is complete. ◀

When all subsets have the same density  $\rho$ , we observe that in fact  $(1 + \rho)(2 - \rho^2) \leq 3$ . Therefore, Theorem 3 follows.

## 5 Fooling Non-Symmetric Confinement Functions

The class of  $t$ -wise confinement functions  $\text{Conf}_n^{\otimes t} \subseteq \{f : [n]^t \rightarrow \{0, 1\}\}$  is defined as  $\text{Conf}_n^{\otimes t} = \{\mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t} \mid S_1, \dots, S_t \subseteq [n]\}$  where  $\mathbf{1}_S(i)$  equals 1 if  $i \in S$  and 0 otherwise. This class of functions is sometimes referred to as cut-tensors or cut-functions. Generally, confinement functions are not symmetric, hence a density-independent total variation bound for this class is not implied by the previous work of [8]. Nevertheless, we show that the class of confinement functions where the sets have equal variances, is  $O(\lambda)$ -fooled by expander random walks regardless of the densities.

We begin with a density-dependent bound which holds for all confinement functions.

► **Proposition 29.** For  $t \geq 1$ , let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $S_1, \dots, S_t$  be a sequence of subsets such that  $S_i$  is of density  $\rho_i$ . Let  $\rho_{\max} = \max_i \rho_i$ . Then,

$$d_{TV}(\mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{RW}_G^t), \mathbf{1}_{S_1} \otimes \dots \otimes \mathbf{1}_{S_t}(\text{Ind}_V^t)) \leq \left(1 + \frac{1 - \rho_{\max}^{t-1}}{1 - \rho_{\max}}\right) \cdot \lambda.$$

**Proof.** First, observe that we may assume  $t \geq 2$ , as for  $t = 1$  the distributions are identical and claim trivially holds. The decomposition  $G = J + \lambda A$  with  $\|A\| \leq 1$  implies that  $\|G - J\| \leq \lambda$ . Also, recall that  $\|JP_i J\| = \rho_i$ . Let  $LHS$  be left-hand size of the inequality in the proposition. Expressing  $LHS$  linear-algebraically, we see that  $LHS = \left| \mathbf{1}^T \prod_{i=1}^t (P_i G) \mathbf{1} - \mathbf{1}^T \prod_{i=1}^t (P_i J) \mathbf{1} \right|$  and

$$\begin{aligned} LHS &\leq \left\| \prod_{i=1}^t (P_i G) - \prod_{i=1}^t (P_i J) \right\| \leq \sum_{k=1}^t \left\| \left( \prod_{j=k+1}^t (P_j G) \right) P_k (G - J) \left( \prod_{j=1}^{k-1} (P_j J) \right) \right\| \\ &\leq \sum_{k=1}^t \|(G - J)\| \left\| \left( \prod_{j=1}^{k-1} (P_j J) \right) \right\| \leq \sum_{k=1}^t \lambda \cdot \prod_{j=1}^{k-1} \rho_j \leq \lambda \left( 1 + \sum_{\ell=0}^{t-2} \rho_{\max}^\ell \right) \\ &= \lambda \cdot \left( 1 + \frac{1 - \rho_{\max}^{t-1}}{1 - \rho_{\max}} \right) \end{aligned} \quad \blacktriangleleft$$

Note that, in particular, the proof implies a  $t\lambda$  bound for all confinement functions. A similar hybrid idea was used in [12] to derive a generalization of the expander mixing lemma for length- $t$  random walks.<sup>3</sup> Proposition 29 shows that when the all the sets are small, say, of density which is bounded from above by some constant  $\alpha$ , the corresponding confinement function is  $O_\alpha(\lambda)$ -fooled.

We proceed with the main result for this section.

**Proof of Theorem 4.** First, observe that we may assume  $t \geq 2$ , as for  $t = 1$  the distributions are identical and claim trivially holds. Let  $LHS$  be left-hand side of the inequality in the proposition. Let  $n = |V|$  and identify  $V$  with  $[n]$  arbitrarily. Let us denote by  $\rho_i$  the density of  $S_i$ , and by  $\sigma_i$  its variance, so that  $\rho_{\max} = \max_i \rho_i$ . Further denote  $\sigma_{\max} = \max_i \sigma_i$ .

We consider two cases according to the relationship between  $\rho_{\max}$  and  $\lambda$ . Assume first that  $\rho_{\max} \leq 2\lambda$ . Applying the upper-bound Proposition 22 we find that

$$\begin{aligned} LHS &= \left| \Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] - \prod_{i=1}^t \rho_i \right| \leq \sqrt{\rho_1 \rho_t} \cdot \prod_{i=1}^{t-1} (\lambda + (1 - \lambda) \sqrt{\rho_i \rho_{i+1}}) \\ &\leq \rho_{\max} (\lambda + \rho_{\max} (1 - \lambda))^{t-1} \leq \rho_{\max} (3\lambda)^{t-1} \leq 3 \cdot \rho_{\max} \cdot \lambda \end{aligned}$$

For the first inequality we observe that both terms inside the absolute value are non-negative, hence the magnitude of their difference is bounded by the maximal one. Additionally, the upper-bound provided by the hitting property as presented in Proposition 22 applies to both terms. Then, we bound  $\lambda + 2\lambda(1 - \lambda) \leq 3\lambda$ . The last inequality holds under our assumption that  $\lambda \leq 1/3$  and  $t \geq 2$ .

<sup>3</sup> In fact their result is more general, as it goes beyond random walk on expander graphs. Their “splittable-mixing lemma” holds for what they call “ $\lambda$ -splittable structures”, which are subsets of  $[n]^t$  that admit certain high-dimensional expansion properties.

### 31:18 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

Now, Assume that  $\rho_{\max} \geq 2\lambda$ . For  $i = 1, \dots, t$ , let  $P_i$  be the  $n \times n$  projection matrix on the set  $S_i$ . That is,  $P_i$  is the diagonal matrix satisfying  $P_i[v, v] = 1$  if  $v \in S_i$  and 0 otherwise. We have the following entry-wise bounds on the flow matrices:  $\tilde{P}_i \leq_{e.w} \begin{pmatrix} \rho_i & \sigma_i \\ \sigma_i & 1 \end{pmatrix}$  where all entries except for the right bottom are equality. To see this, consider that

$$\|\Pi_0 P_i \Pi_0\| = \|J P_i J\| = \|\mathbf{1} \mathbf{1}^T P_i \mathbf{1} \mathbf{1}^T\| = |\mathbf{1}^T P_i \mathbf{1}| = \rho_i.$$

Moreover, for  $b \in \{0, 1\}$  we have

$$\|\Pi_b P_i \Pi_0\| = \|\Pi_b P_i \mathbf{1} \mathbf{1}^T\| = \|\Pi_b P_i \mathbf{1}\| \cdot \|\mathbf{1}\| = \|\Pi_b P_i \mathbf{1}\|.$$

Since  $\|P_i \mathbf{1}\| = \sqrt{\sum_{i \in S_i} \frac{1}{n}} = \sqrt{\rho_i}$ , we have

$$\|\Pi_1 P_i \Pi_0\| = \|\Pi_1 P_i \mathbf{1}\| = \sqrt{\|P_i \mathbf{1}\|^2 - \|\Pi_0 P_i \mathbf{1}\|^2} = \sqrt{\rho_i(1 - \rho_i)} = \sigma_i.$$

By symmetry we also have  $\|\Pi_1 P_i \Pi_0\| = \sigma_i$ . Finally, we bound  $\|\Pi_1 P_i \Pi_1\| \leq \|\Pi_1\|^2 \|P_i\| \leq 1$ .

Let  $\sigma_{\max} = \max_i \sigma_i$ , and recall that by assumption  $\sigma_{\max} = \sqrt{\rho_{\max}(1 - \rho_{\max})}$ . Through utilizing the submultiplicativity of the flow operator (Claim 11) and Example 10, we find that  $\tilde{G}P_i \leq_{e.w} \begin{pmatrix} \rho_i & \sigma_i \\ \lambda \sigma_i & \lambda \end{pmatrix} \leq_{e.w} A$  for  $A \stackrel{\text{def}}{=} \begin{pmatrix} \rho_{\max} & \sigma_{\max} \\ \lambda \sigma_{\max} & \lambda \end{pmatrix}$ . Now, expressing the total variation distance linear algebraically, we have

$$\begin{aligned} LHS &= \left| \Pr_{(v_1, \dots, v_t) \sim \text{RW}_G^t} [\forall i v_i \in S_i] - \prod_{i=1}^t \rho_i \right| = \left| \mathbf{1}^T \prod_{i=2}^t (P_i G) P_1 \mathbf{1} - \mathbf{1}^T \prod_{i=2}^t (P_i J) P_1 \mathbf{1} \right| \\ &= \left| \mathbf{1}^T \prod_{i=1}^t (G P_i) \mathbf{1} - \mathbf{1}^T \prod_{i=1}^t (J P_i) \mathbf{1} \right| = \left| \mathbf{1}^T \prod_{i=1}^t (G P_i) \mathbf{1} - \mathbf{1}^T \prod_{i=1}^t (\Pi_0 G P_i) \mathbf{1} \right| \\ &\leq \left( \prod_{i=1}^t \tilde{G}P_i \right) [0, 0] - \left( \prod_{i=1}^t \tilde{G}P_i [0, 0] \right) \\ &\leq A^t [0, 0] - (A[0, 0])^t \end{aligned}$$

Where the first inequality is by Lemma 16, and the second is by Lemma 17 Using Lemma 15 we obtain the bound

$$\begin{aligned} A^t [0, 0] - (A[0, 0])^t &\leq \frac{\lambda \rho_{\max} \sigma_{\max}^2}{\rho_{\max} - \lambda} \cdot \sum_{k=0}^{t-2} \rho_{\max}^k \left( \rho_{\max} + \frac{\lambda \sigma_{\max}^2}{\rho_{\max} - \lambda} \right)^{t-k-2} \\ &= \frac{\lambda \rho_{\max}^2 (1 - \rho_{\max})}{\rho_{\max} - \lambda} \cdot \sum_{k=0}^{t-2} \rho_{\max}^k \left( \rho_{\max} + \frac{\lambda \rho_{\max} (1 - \rho_{\max})}{\rho_{\max} - \lambda} \right)^{t-k-2} \\ &\leq 2\rho_{\max} \cdot \lambda \cdot (1 - \rho_{\max}) \sum_{k=0}^{t-2} \rho_{\max}^k (\rho_{\max} + \rho_{\max} (1 - \rho_{\max}))^{t-k-2} \\ &\leq 2\rho_{\max} \cdot \lambda \cdot (1 - \rho_{\max}) \sum_{k=0}^{\infty} \rho_{\max}^k = 2\rho_{\max} \cdot \lambda, \end{aligned}$$

where in the second inequality we have used that  $\rho_{\max}/(\rho_{\max} - \lambda) \leq 2$  and  $\lambda/(\rho_{\max} - \lambda) \leq 1$  under our assumption that  $\rho_{\max} \geq 2\lambda$ .  $\blacktriangleleft$

## 6 Fooling The Sum Function modulo $d$

For integers  $d \geq 2$ ,  $d' \geq 2$  and  $t \geq 1$ , define the function  $\text{Sum}_{d',d} : \mathbb{Z}_{d'}^t \rightarrow \mathbb{Z}_d$  as  $\text{Sum}_{d',d}(a_1, \dots, a_t) = \sum_{i=1}^t a_i \bmod d$ . Given the insignificance of  $d'$  within this context, we will simplify our notation by omitting it, using only  $\text{Sum}_d$ .

In this section we use the flow framework to prove a bias-independent  $O(\sqrt{d} \cdot \lambda)$  total variation bound for  $\text{Sum}_d$ . We also prove a bias amplification result (Lemma 33) from which an  $O(\sqrt{d} \cdot c^t)$  total variation bound can be derived using Lemma 8, where  $c < 1$  is a parameter that depends on the bias of the labeling and  $\lambda$ . [8] obtains a similar bound, which they derive from their results on permutation branching programs.

► **Remark 30.** While characters of  $\mathbb{Z}_d$  are formally defined on values in  $\mathbb{Z}_d$ , throughout this section, we simplify notation by using  $\chi(a)$  for an arbitrary integer  $a$  and character  $\chi$  of  $\mathbb{Z}_d$ , to mean  $\chi(a \bmod d)$ .

### 6.1 Bias Amplification

We begin our discussion with the Boolean case.

► **Definition 31** (bias over  $\mathbb{Z}_2$ ). *The bias of a labeling  $\text{val} : [n] \rightarrow \{0, 1\}$  is defined as  $\text{bias}(\text{val}) \stackrel{\text{def}}{=} |\mathbb{E}_{i \in [n]} (-1)^{\text{val}(i)}|$ .*

The distribution over  $\{0, 1\}$  obtained by sampling a uniformly random element of  $[n]$  and outputting its label is  $\text{bias}(\text{val})$ -biased. However, the distribution obtained by taking  $t$  uniformly random samples from  $[n]$  and computing the parity of the corresponding labels is only  $\text{bias}(\text{val})^t$ -biased. That is, the bias decreases exponentially with  $t$ . To see this, note that

$$\left| \mathbb{E}_{(i_1, \dots, i_t) \in [n]^t} (-1)^{\sum_{j=1}^t \text{val}(i_j)} \right| = \left| \prod_{j=1}^t \mathbb{E}_{i_j \in [n]} (-1)^{\text{val}(i_j)} \right| = \text{bias}(\text{val})^t.$$

It has been observed in [13] that this bias reducing construction can be derandomized by taking length- $(t-1)$  expander random walks on  $[n]$  rather than independent samples. In this case, it is shown that the bias of the resulting distribution is at most  $(\text{bias}(\text{val}) + \lambda)^{\lfloor t/2 \rfloor}$ , where  $\lambda$  is the expansion parameter of the graph. In [13], this property is called *parity sampling*, and it follows that expander random walks are *good parity samplers*. This observation is a key part of the breakthrough construction of almost optimal  $\varepsilon$ -balanced codes [13].

In the context of the sum function modulo  $d$ , we allow labelings with a larger alphabet size. It is therefore natural to ask whether the bias amplification phenomenon extends to  $\mathbb{Z}_d$  where  $d > 2$ . Observe that the bias of a labeling  $\text{val} : [n] \rightarrow \{0, 1\}$  is simply the inner product of the distribution induced by the labeling with the non-trivial character of  $\mathbb{Z}_2$ . This notion extends naturally to characters of  $\mathbb{Z}_d$  as follows.

► **Definition 32** (bias over  $\mathbb{Z}_d$ ). *For integers  $d \geq 2$  and  $d' \geq 2$ , the bias of a labeling  $\text{val} : [n] \rightarrow \mathbb{Z}_{d'}$  with respect to a character  $\chi$  of  $\mathbb{Z}_d$  is defined as  $\text{bias}_\chi(\text{val}) \stackrel{\text{def}}{=} |\mathbb{E}_{i \in [n]} \chi(\text{val}(i))|$ .*

The same argument as before shows that for any character  $\chi$  of  $\mathbb{Z}_d$ , taking  $t$  independent samples from  $[n]$  and outputting the sum of their labels modulo  $d$  yields a distribution on  $\mathbb{Z}_d$  with bias  $\text{bias}_\chi(\text{val})^t$  with respect to the character  $\chi$ . Moreover, we prove that replacing the independent samples by length  $t-1$  random walk on a  $\lambda$ -expander graph obtains a distribution on  $\mathbb{Z}_d$  with bias at most  $(\text{bias}_\chi(\text{val}) + \lambda)^{\lfloor t/2 \rfloor}$  with respect to the character  $\chi$ . (In fact, the bound is slightly better, as here it may be larger than 1). In other words, expander random walks are *good character samplers*. This fact has also been independently observed in [11].

## 31:20 The Expander Hitting Property When the Sets Are Arbitrarily Unbalanced

► **Lemma 33** (Bias Amplification). *For integers  $t \geq 2$ ,  $d \geq 2$ , and  $d' \geq 2$ , let  $G = (V, E)$  be a  $\lambda$ -expander, and let  $\text{val} : V \rightarrow \mathbb{Z}_{d'}$  be any labeling. Let  $\chi$  be a character of  $\mathbb{Z}_d$ . Then*

$$\left| \mathbb{E}_{(v_1, \dots, v_t) \sim \text{RW}_G^t} \chi \left( \sum_{i=1}^t \text{val}(v_i) \right) \right| \leq ((1 - \lambda)^2 \cdot \text{bias}_\chi(\text{val}) + 2\lambda(1 - \lambda) + \lambda^2)^{\lfloor \frac{t}{2} \rfloor}, \quad (4)$$

where  $\text{bias}_\chi(\text{val}) = \left| \mathbb{E}_{i \in [n]} \chi(\text{val}(i)) \right|$  is the bias of  $\text{val}$  with respect to  $\chi$ .

The proof for Lemma 33 appears in the full version of this paper [14]

### 6.2 Bias-independent bound using the flow framework

When the bias is bounded by a constant, the bias amplification property implies that the distributions  $\text{Sum}_d(\text{val}(\text{Ind}_V^t))$  and  $\text{Sum}_d(\text{val}(\text{RW}_G^t))$  are highly unbiased, with bias which is exponentially small in  $t$ . Applying the triangle inequality to the XOR-Lemma (Lemma 8), we see that the total variation distance between these distributions is bounded by the sum of the biases of each distribution with respect to a worst-case non-trivial character. As such, it is decreasing exponentially fast with  $t$  as well. However, this argument hinges on the assumption that the given labeling is balanced. If we have, say,  $\text{bias}_\chi(\text{val}) = 1 - 1/t$  for some non-trivial character  $\chi$  of  $\mathbb{Z}_d$ , the bias amplification argument is insufficient for an effective total variation bound. This constitutes the primary reason why earlier works such as [5] and [4], which rely heavily on the bias-amplification property, result in total variation bounds that are bias-dependent.

Next, we use the flow framework to obtain an  $O(\sqrt{d} \cdot \lambda)$  bias independent bound, similar to that of [8]. The proof in this case is arguably simpler than the more general case in [8], which applies for all small-width permutation branching programs.

**Proof of Theorem 6.** First, observe that we may assume  $t \geq 2$ , as for  $t = 1$  the distributions are identical, and the claim trivially holds. Let  $LHS$  be left-hand side of the inequality in the theorem. Let  $n = |V|$  and identify  $V$  with  $[n]$  arbitrarily. Observe that in order to obtain a total variation bound, it suffices to bound the maximum bias of the difference between the two distributions. Indeed, by the XOR-Lemma (Lemma 8 )

$$LHS \leq \frac{\sqrt{d}}{2} \cdot \max_{\chi \in \widehat{\mathbb{Z}}_d} |\langle \chi, \text{Sum}_d(\text{val}(\text{RW}_G^t)) - \text{Sum}_d(\text{val}(\text{Ind}_V^t)) \rangle|$$

We fix a character  $\chi$  of  $\mathbb{Z}_d$  that attains the maximum. Let  $\mu = \text{bias}_\chi(\text{val})$  be the bias of the labeling  $\text{val}$  with respect to  $\chi$ . We consider two cases according to the relation between  $\mu$  and  $\lambda$ . To begin, let us assume that  $\mu \leq 3\lambda$ . In that case,

$$\begin{aligned} LHS &\leq \frac{\sqrt{d}}{2} \left| \mathbb{E}_{v=(v_1, \dots, v_t) \sim \text{RW}_G^t} \chi \left( \sum_{i=1}^t \text{val}(v_i) \right) - \mathbb{E}_{v \sim \text{Ind}_V^t} \chi \left( \sum_{i=1}^t \text{val}(v_i) \right) \right| \\ &\leq \sqrt{d} \cdot ((1 - \lambda)^2 \cdot \mu + 2\lambda(1 - \lambda) + \lambda^2)^{\lfloor \frac{t}{2} \rfloor} \\ &\leq \sqrt{d} \cdot (3(1 - \lambda)^2 \cdot \lambda + 2\lambda(1 - \lambda) + \lambda^2)^{\lfloor \frac{t}{2} \rfloor} \leq \sqrt{d} \cdot (5\lambda)^{\lfloor \frac{t}{2} \rfloor} \leq 5\sqrt{d} \cdot \lambda, \end{aligned} \quad (5)$$

where the second inequality is implied by the triangle inequality and the bias amplification property established in the previous subsection. The last inequality holds under our assumption that  $t \geq 2$  and  $\lambda \leq 1/6$ .



Now, let us assume that  $\mu \geq 3\lambda$ . Instead of applying the triangle inequality on the second line of Equation 5, we express it linear algebraically. We then give entry-wise bounds for the flow matrices of the involved linear operators. Let  $P$  be the  $n \times n$  diagonal matrix  $P = \text{diag}(\chi(\text{val}(1)), \dots, \chi(\text{val}(n)))$ . We have the following entry-wise bounds on the flow matrix:  $\tilde{P} \leq_{e.w} \begin{pmatrix} \mu & \sqrt{1-\mu^2} \\ \sqrt{1-\mu^2} & 1 \end{pmatrix}$  where all entries except for the right bottom are equality. To see this, note that

$$\tilde{P}[0,0] = \|\Pi_0 P \Pi_0\| = \|J P J\| = \|\mathbf{1} \mathbf{1}^T P \mathbf{1} \mathbf{1}^T\| = |\mathbf{1}^T P \mathbf{1}| = \mu.$$

Moreover, we see that for  $b \in \{0, 1\}$ ,

$$\|\Pi_b P \Pi_0\| = \|\Pi_b P \mathbf{1} \mathbf{1}^T\| = \|\Pi_b P \mathbf{1}\| \cdot \|\mathbf{1}^T\| = \|\Pi_b P \mathbf{1}\|.$$

Now, since  $P$  is unitary,

$$\tilde{P}[0,0]^2 + \tilde{P}[1,0]^2 = \|\Pi_0 P \mathbf{1}\|^2 + \|\Pi_1 P \mathbf{1}\|^2 = \|P \mathbf{1}\|^2 = \|\mathbf{1}\|^2 = 1$$

By symmetry we conclude that  $\tilde{P}[1,0] = \tilde{P}[0,1] = \sqrt{1-\mu^2}$ . Finally, we bound  $\tilde{P}[1,1] = \|\Pi_1 P \Pi_1\| \leq \|\Pi_1\|^2 \|P\| \leq 1$ . By submultiplicativity of the flow operator (Claim 11) and Example 10, We see that  $\widetilde{GP} \leq_{e.w} A$  for  $A \stackrel{\text{def}}{=} \begin{pmatrix} \mu & \sqrt{1-\mu^2} \\ \lambda \sqrt{1-\mu^2} & \lambda \end{pmatrix}$ . Now, Let us pick up Equation 5 after the first inequality. Expressing the bias linear-algebraically,

$$\left| \mathbb{E}_{v \sim \text{RW}_G^t} \chi \left( \sum_{i=1}^t \text{val}(v_i) \right) - \mathbb{E}_{v \sim \text{Ind}_V^t} \chi \left( \sum_{i=1}^t \text{val}(v_i) \right) \right| = |\mathbf{1}^T (PG)^{t-1} P \mathbf{1} - \mathbf{1}^T (PJ)^{t-1} P \mathbf{1}|$$

and,

$$\begin{aligned} |\mathbf{1}^T (PG)^{t-1} P \mathbf{1} - \mathbf{1}^T (PJ)^{t-1} P \mathbf{1}| &= |\mathbf{1}^T (GP)^{t-1} \mathbf{1} - \mathbf{1}^T (JP)^{t-1} \mathbf{1}| \\ &= |\mathbf{1}^T (GP)^{t-1} \mathbf{1} - \mathbf{1}^T (\Pi_0 G P)^{t-1} \mathbf{1}| && (\Pi_0 G = J) \\ &\leq (\widetilde{GP})^{t-1}[0,0] - (\widetilde{GP})^{t-1}[0,0] && (\text{Lemma 16}) \\ &\leq A^{t-1}[0,0] - (A[0,0])^{t-1} && (\text{Lemma 17}) \end{aligned}$$

Applying Lemma 13 we obtain the bound

$$\begin{aligned} A^t[0,0] - (A[0,0])^t &\leq \frac{\lambda \mu (1-\mu^2)}{\mu-\lambda} \sum_{k=0}^{t-2} \mu^k \left( \mu + \frac{\lambda(1-\mu^2)}{\mu-\lambda} \right)^{t-k-2} \\ &\leq \frac{3}{2} \cdot \lambda (1-\mu^2) \sum_{k=0}^{t-1} \mu^k \left( \mu + \frac{1}{2} \cdot (1-\mu^2) \right)^{t-k-1} \leq \frac{3}{2} \cdot \lambda \cdot (1+\mu)(1-\mu) \sum_{k=0}^{\infty} \mu^k \\ &\leq 3 \cdot \lambda. \end{aligned}$$

where the second inequality holds as our assumption  $\mu \geq 3\lambda$  implies that  $\mu/(\mu-\lambda) \leq 3/2$  and  $\lambda/(\mu-\lambda) \leq 1/2$ . In the third inequality we have used that  $\mu + \frac{1}{2} \cdot (1-\mu^2) \leq 1$ . Overall, we have  $d_{TV}(\text{Sum}_d(\text{val}(\text{RW}_G^t)), \text{Sum}_d(\text{val}(\text{Ind}_V^t))) < 5\sqrt{d} \cdot \lambda$  in all cases, and the proof is complete.  $\blacktriangleleft$

---

**References**

---

- 1 Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in logspace. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 132–140, 1987.
- 2 Noga Alon, Uriel Feige, Avi Wigderson, and David Zuckerman. Derandomized graph products. *Computational Complexity*, 5:60–75, 1995.
- 3 C. Beck. Chernoff bounds for expander walks. <https://www.ias.edu/video/csdlm/2015/0310-ChristopherBeck>, 2015. A recording of a lecture, given at the Institute for Advanced Study, Princeton, USA.
- 4 Gil Cohen, Dor Minzer, Shir Peleg, Aaron Potechin, and Amnon Ta-Shma. Expander random walks: The general case and limitations. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 5 Gil Cohen, Noam Peri, and Amnon Ta-Shma. Expander random walks: A fourier-analytic approach. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1643–1655, 2021.
- 6 David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.
- 7 Oded Goldreich. Three XOR-lemmas – An exposition. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 248–272, 2011.
- 8 Louis Golowich and Salil Vadhan. Pseudorandomness of expander random walks for symmetric functions and permutation branching programs. In *37th Computational Complexity Conference (CCC 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 9 Alexander D Healy. Randomness-efficient sampling within nc. *Computational Complexity*, 17:3–37, 2008.
- 10 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
- 11 Akhil Jalan and Dana Moshkovitz. Near-optimal cayley expanders for abelian groups. *arXiv preprint*, 2021. [arXiv:2105.01149](https://arxiv.org/abs/2105.01149).
- 12 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.
- 13 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, 2017.
- 14 Amnon Ta-Shma and Ron Zadiario. The expander hitting property when the sets are arbitrarily unbalanced. *Electronic Colloquium on Computational Complexity (ECCC)*, TR24-118, 2024. URL: <https://eccc.weizmann.ac.il/report/2024/118>.
- 15 Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

# Near-Linear Time Samplers for Matroid Independent Sets with Applications

Xiaoyu Chen ✉

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

Heng Guo ✉

School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, UK

Xinyuan Zhang ✉

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

Zongrui Zou ✉

State Key Laboratory for Novel Software Technology, New Cornerstone Science Laboratory, Nanjing University, 163 Xianlin Avenue, Nanjing, Jiangsu Province, China

---

## Abstract

We give a  $\tilde{O}(n)$  time almost uniform sampler for independent sets of a matroid, whose ground set has  $n$  elements and is given by an independence oracle. As a consequence, one can sample connected spanning subgraphs of a given graph  $G = (V, E)$  in  $\tilde{O}(|E|)$  time, whereas the previous best algorithm takes  $O(|E||V|)$  time. This improvement, in turn, leads to a faster running time on estimating all-terminal network reliability. Furthermore, we generalise this near-linear time sampler to the random cluster model with  $q \leq 1$ .

**2012 ACM Subject Classification** Theory of computation → Random walks and Markov chains

**Keywords and phrases** Network reliability, Random cluster model, Matroid, Bases-exchange walk

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.32

**Category** RANDOM

**Funding** *Heng Guo*: 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).

**Acknowledgements** We would like to thank the hospitality of NII Shonan meeting No. 186, where some of the discussion took place.

## 1 Introduction

Let  $\mathcal{M} = ([n], \mathcal{I})$  be a matroid of rank  $r$  and  $\lambda \in \mathbb{R}_{>0}^n$  be the external fields (namely weights for the ground set elements). Denote its set of bases by  $\mathcal{B} = \mathcal{B}(\mathcal{M})$ , and by  $\mathcal{I} = \mathcal{I}(\mathcal{M})$  the set of independent sets. Suppose that we want to sample a random base  $B \in \mathcal{B}$  from the following distribution:

$$\forall B \in \mathcal{B}, \quad \mu_{\mathcal{B}, \lambda}(B) \propto \prod_{i \in B} \lambda_i.$$

There is a natural Markov chain, namely the bases-exchange walk (also known as the down-up walk) [8], that converges to the distribution above. Anari, Liu, Oveis Gharan, and Vinzant [2] showed that this chain mixes in polynomial time. Subsequently, Cryan, Guo, and Mousa [7] and a follow up work by Anari, Liu, Oveis Gharan, Vinzant, and Vuong [3] refined the mixing time to the optimal  $O(r \log r)$ .



© Xiaoyu Chen, Heng Guo, Xinyuan Zhang, and Zongrui Zou;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 32; pp. 32:1–32:12



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

In this work, we focus on another important distribution associated with the matroid  $\mathcal{M}$ , namely, the distribution  $\mu_{\mathcal{M},\lambda}$  over the independent sets of  $\mathcal{M}$ :

$$\forall S \in \mathcal{I}, \quad \mu_{\mathcal{M},\lambda}(S) \propto \prod_{i \in S} \lambda_i. \quad (1)$$

As suggested by the previous work [3], in order to sample from  $\mu_{\mathcal{M},\lambda}$ , we may construct another matroid  $\mathcal{M}_{\mathcal{I}}$  so that there is a one-to-one correspondence between the bases of  $\mathcal{M}_{\mathcal{I}}$  and the independent sets of  $\mathcal{M}$ . Therefore, we may use the bases-exchange walk on  $\mathcal{M}_{\mathcal{I}}$  to approximately generate samples from  $\mu_{\mathcal{M},\lambda}$  within  $O(n \log n)$  steps.

However, an efficient implementation of the bases-exchange walk on  $\mathcal{M}_{\mathcal{I}}$  is far from trivial. Note that the input matroid is usually given succinctly. For example, for a graphic matroid, it is sufficient to input the associated graph, rather than a list of all the bases or independent sets, which can be exponentially large in the size of the graph. The most common and standard way is to give the matroid by an independence oracle  $\mathcal{O}_I$ . Upon receiving a subset of elements, the independence oracle  $\mathcal{O}_I$  returns whether or not the set is independent. Given  $\mathcal{O}_I$ , the naïve implementation of the bases-exchange walk requires  $O(n)$  oracle calls per step. Depending on the application, there may be even further cost of implementing the oracle. This prevents us from getting a near-linear time sampler for many potential applications.

To get a faster algorithm, Anari, Liu, Oveis Gharan, Vinzant, and Vuong considered a different oracle  $\mathcal{O}'$  [3]. One can query  $\mathcal{O}'$  a subset  $S \subseteq [n]$  with the promise that  $S$  contains at most one circuit. If a circuit exists,  $\mathcal{O}'$  will output a uniformly random element of the circuit. In [3], it is showed that there is a sampling algorithm for  $\mu_{\mathcal{M},\lambda}$  using  $O(n \log n)$  queries in total of  $\mathcal{O}'$ . However the downside is that this oracle  $\mathcal{O}'$  is typically more difficult to implement than the independence oracle  $\mathcal{O}_I$ . In our applications to be discussed later (network reliability and the random cluster model), the independence oracle  $\mathcal{O}_I$  can be implemented in logarithmic time (at least in an amortized sense), whereas it appears to require at least linear time to implement  $\mathcal{O}'$ . A straightforward implementation of  $\mathcal{O}'$  requires  $O(r)$  calls to  $\mathcal{O}_I$ , where  $r$  is the rank of the matroid  $\mathcal{M}$ . This leads to a sampling algorithm using  $O(rn \log n)$  oracle calls to  $\mathcal{O}_I$ . Note that the rank  $r$  is often not a constant and can be as large as  $\Omega(n)$ .

We give a sampler which requires  $O((1 + \lambda_{\max})n \log(n/\varepsilon))$  oracle calls in expectation to  $\mathcal{O}_I$ , where  $\lambda_{\max} := \max_{i \in [n]} \lambda_i$  and  $\varepsilon$  is the sampling error. This improves the previously best  $O(rn \log n)$  running time [3]. We use the *total variation distance* (TV distance) to measure the distance between two distributions  $\mu$  and  $\nu$  over a finite space  $\Omega$ , defined by  $D_{\text{TV}}(\mu, \nu) := \frac{1}{2} \sum_{X \in \Omega} |\mu(X) - \nu(X)|$ .

► **Theorem 1.** *Equipped with the independence oracle  $\mathcal{O}_I$  of a matroid  $\mathcal{M} = ([n], \mathcal{I})$ , there exists an algorithm that takes external fields  $\lambda \in \mathbb{R}_{>0}^n$  and  $\varepsilon \in (0, 1)$  as inputs, and outputs a random set  $S \in \mathcal{I}$  satisfying  $D_{\text{TV}}(\mu_{\mathcal{M},\lambda}, S) \leq \varepsilon$ . In expectation, it runs in  $O((1 + \lambda_{\max})n \log(n/\varepsilon)(\log n + t_{\mathcal{O}_I}))$  time, where  $t_{\mathcal{O}_I}$  is the time to answer a query by the independence oracle  $\mathcal{O}_I$  and  $\lambda_{\max} := \max_{i \in [n]} \lambda_i$ .*

The proof of Theorem 1 is given in Section 3.

► **Remark 2.** In particular, instead of the independence oracle  $\mathcal{O}_I$ , for our algorithm in Theorem 1, it suffices to have a data structure maintaining a set  $S \subseteq [n]$  which supports:

- to *insert* an element to  $S$ ;
- to *delete* an element from  $S$ ;
- and to *query* if  $S \in \mathcal{I}$ .

Given such a data structure,  $t_{\mathcal{O}_I}$  in Theorem 1 can be substituted by the worst case or amortized running time of these operations.

The crux of Theorem 1 is a fast implementation of the transition step of the bases-exchange chain for  $\mathcal{M}_{\mathcal{T}}$ . Note that the transition of a Markov chain is in itself yet another sampling problem. We design a rejection sampling procedure for this latter sampling task. There is a constant upper bound for the rejection probability (see Lemma 14), guaranteeing a success with high probability in logarithmic trials.

A consequence of our algorithm is a faster approximation algorithm for the *all-terminal network reliability*. The problem is defined as follows. Given a connected undirected graph  $G = (V, E)$  and failure probabilities  $\mathbf{p} \in \mathbb{R}_{>0}^E$ , the all-terminal network reliability  $Z_{\text{rel}}(G, \mathbf{p})$  is the probability that the graph is connected if each edge  $e$  fails (i.e. is removed) independently with probability  $p_e$ . Formally, for  $S \subseteq E$ , let

$$\text{wt}(S) := \mathbf{1}[G[E \setminus S] \text{ is connected}] \cdot \prod_{e \in S} p_e \prod_{f \in E \setminus S} (1 - p_f), \quad (2)$$

where  $G[E \setminus S]$  is the spanning subgraph of  $G$  on  $E \setminus S$ . Then the reliability of the network is

$$Z_{\text{rel}}(G, \mathbf{p}) := \sum_{S \subseteq E} \text{wt}(S).$$

By standard techniques [16, 20, 17], estimating  $Z_{\text{rel}}(G, \mathbf{p})$  can be reduced to approximate sampling of the (weighted) distribution of connected spanning subgraphs:

$$\forall S \subseteq E, \quad \mu_{G, \mathbf{p}}^{\text{NR}}(S) \propto \text{wt}(S). \quad (3)$$

The study of the computational complexity of network reliability was initiated by Valiant [21]. Exact evaluation of the all-terminal version is known to be  $\#\mathbf{P}$ -hard [15, 19]. Guo and Jerrum [11] gave the first fully polynomial-time randomized approximate scheme (FPRAS) using the *partial rejection sampling* framework [12]. This algorithm samples from  $\mu_{G, \mathbf{p}}^{\text{NR}}$  in  $O(|E| + \frac{p_{\max}|V||E|}{1-p_{\max}})$  time in expectation [10] where  $p_{\max} := \max_i p_i$  is the maximum failure probability, and this bound is tight for the technique. It is also worth mentioning that, using the result in [3] directly, it is possible to get an  $\tilde{O}(|V||E|)$  time sampler, whose running time is of roughly the same order as the partial rejection sampling algorithm.

Using Theorem 1, we obtain an  $\tilde{O}(|V|)$  speed-up to sample from  $\mu_{G, \mathbf{p}}^{\text{NR}}$ . This gives the first near-linear time sampler for connected spanning subgraphs.

► **Corollary 3.** *Let  $G = (V, E)$  be a connected graph with  $n$  vertices and  $m$  edges. Let  $\mathbf{p} \in (0, 1)^E$  be the failure probabilities for edges. There is an algorithm that takes  $G$ ,  $\mathbf{p}$  and  $\varepsilon \in (0, 1)$  as input, and outputs a random subset  $S \subseteq E$  such that  $D_{\text{TV}}(\mu_{G, \mathbf{p}}^{\text{NR}}, S) \leq \varepsilon$  in  $O\left(\frac{m(\log^3 n + \log \frac{1}{\varepsilon})}{1-p_{\max}}\right)$  time in expectation, where  $p_{\max} := \max_{e \in E} p_e$ .*

**Proof.** Let  $\Omega := \{S \subseteq E \mid \mu_{G, \mathbf{p}}^{\text{NR}}(S) > 0\}$  be the support of  $\mu_{G, \mathbf{p}}^{\text{NR}}$ . Recall that  $\mu_{G, \mathbf{p}}^{\text{NR}}(S) > 0$  if and only if  $G[E \setminus S]$  is connected. This means that there is a spanning tree  $T$  of  $G$  contained in  $E \setminus S$ . Note that spanning trees are bases of the graphic matroid  $\mathcal{M}_G$  of a graph  $G$ . Hence, let  $\mathcal{M}_{\text{NR}} := (E, \Omega)$ , it holds that  $\mathcal{M}_{\text{NR}}$  is the dual matroid of  $\mathcal{M}_G$ , namely the co-graphic matroid. This also means that  $\mu_{G, \mathbf{p}}^{\text{NR}} = \mu_{\mathcal{M}, \lambda}$  for  $\mathcal{M} = \mathcal{M}_{\text{NR}}$  and  $\lambda_e = \frac{p_e}{1-p_e}, \forall e \in E$  as defined in (1).

It remains to implement the independence oracle efficiently. For this we use dynamic data structures for connectivity of graphs, which is a topic that has been extensively studied. For  $\mathcal{M}_{\text{NR}}$ , as in Remark 2, we implement the independence oracle  $\mathcal{O}_I$  with amortized cost  $t_{\mathcal{O}_I} = O(\log^2 n)$  by using the data structure in [22, Section 3] directly. The corollary follows by combining this with Theorem 1. ◀

As we can see in the proof, the advantage of our algorithm is that the independence oracle can be implemented in amortized logarithmic time. In contrast, it requires at least linear time to implement the oracle  $\mathcal{O}'$  of Anari, Liu, Oveis Gharan, Vinzant, and Vuong [3]. Our sampling algorithm in Theorem 1 calls the independence oracle only a near-linear number of times, which is crucial to obtain the overall near-linear running time.

Using the counting to sampling reduction in [10], Corollary 3 implies an FPRAS that outputs an  $(1 \pm \varepsilon)$ -approximation of  $Z_{\text{rel}}(G, \mathbf{p})$  in time  $O\left(\frac{mn \log^4(n)}{\varepsilon^2(1-p_{\max})} \log \frac{1}{1-p_{\max}}\right)$ . As before, this improves the previous best running time by a factor of  $\tilde{O}(n)$ . We also note that the running time in Corollary 3 is linear in  $(1 - p_{\max})^{-1}$ . This factor comes from our rejection sampling implementation of the down-up walk, whereas the naïve implementation of the down-up walk has logarithmic dependence. On the other hand, in most applications,  $1 - p_{\max} = \Omega(1)$ . For example, for the uniform distribution over connected spanning subgraphs,  $p_e = 1/2$  for all  $e$ . Moreover, in the simulated annealing counting to sampling reduction [10], we do not need to consider  $p_e$  larger than  $p_{\max}$ . Thus, in this reduction, there is no extra slowdown due to this linear dependence on  $1 - p_{\max}$ . In any case, we leave improving this dependence for near-linear time samplers as an open problem.

The distribution  $\mu_{G, \mathbf{p}}^{\text{NR}}$  in (3) is a special case of the random cluster model on the graph  $G$  with parameter  $q = 0$  [9]. More generally, for a matroid  $\mathcal{M} = ([n], \mathcal{I})$  with a rank function  $\text{rk}(\cdot)$ , the random cluster model with parameter  $q \geq 0$  and external fields  $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^n$  is defined as follows: for  $S \subseteq X$ ,

$$\pi_{RC, q, \boldsymbol{\lambda}}(S) \propto q^{-\text{rk}(S)} \prod_{x_i \in S} \lambda_i. \quad (4)$$

For  $q = 0$ , the support of the distribution in (4) must have the highest rank. For a graphic matroid over a graph  $G$ , this means that  $G[S]$  must be connected, namely  $S$  in (4) corresponds to  $E \setminus S$  in  $\mu_{G, \mathbf{p}}^{\text{NR}}$ .

We also extend our near-linear time sampler to random cluster models with  $q \leq 1$ . Note that the rank of  $S$  plays an important role in the distribution in  $\pi_{RC, q, \boldsymbol{\lambda}}$  (4). Thus, instead of the independence oracle, here we use a rank oracle, which upon a query  $S$  returns the rank of  $S$ .

► **Theorem 4.** *Let  $0 \leq q \leq 1$  be a parameter. Equipped with the rank oracle  $\mathcal{O}_r$  of a matroid  $\mathcal{M} = ([n], \mathcal{I})$ , there exists an algorithm that takes external fields  $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^n$  and  $\varepsilon \in (0, 1)$  as inputs, and outputs a random set  $S$  such that  $D_{\text{TV}}(\pi_{RC, q}, S) \leq \varepsilon$ . It runs in  $O\left((1 + \lambda_{\min}^{-1})n \log(n/\varepsilon)(\log n + t_{\mathcal{O}_r})\right)$  time in expectation, where  $t_{\mathcal{O}_r}$  is the time to answer a query by the rank oracle  $\mathcal{O}_r$  and  $\lambda_{\min} := \min_{i \in [n]} \lambda_i$ .*

Theorem 4 is proved in Section 4.

Similar to Remark 2, it suffices to replace the rank oracle by a data structure that supports insertion/deletion of elements and query if the rank changes after removing an element. For graphs, the rank oracle, once again, can be implemented using the data structure in [22]. This is because  $\text{rk}(S) = |V| - 1 + \kappa(E) - \kappa(S)$ , where  $\kappa(S)$  is the number of connected components in  $G[S]$ . Thus the rank change query in graphs is exactly the same as asking if  $u$  and  $v$  are connected after removing an edge  $(u, v)$ . The amortized cost of using this data structure is  $O(\log^2 n)$ .

► **Corollary 5.** *Let  $G = (V, E)$  be a graph with  $n$  vertices and  $m$  edges. Let  $q \leq 1$  be a parameter and  $\boldsymbol{\lambda} \in \mathbb{R}_{>0}^E$  be the external fields on edges. There is an algorithm that takes  $G$ ,  $\boldsymbol{\lambda}$  and  $\varepsilon \in (0, 1)$  as input, and outputs a random subset  $S \subseteq E$  such that  $D_{\text{TV}}(\pi_{RC, q}, S) \leq \varepsilon$  in  $O\left((1 + \lambda_{\min}^{-1})m(\log^3 n + \log \frac{1}{\varepsilon})\right)$  time in expectation, where  $\lambda_{\min} := \min_{i \in [n]} \lambda_i$ .*

## 2 Preliminaries

### 2.1 Matroid

Matroid is an abstract combinatorial structure that generalizes the notion of linear independence. It is usually specified by a pair  $\mathcal{M} = (U, \mathcal{I})$  where  $U$  is a ground set and  $\mathcal{I} \subseteq 2^U$  is a collection of subsets of  $U$ . The subsets in  $\mathcal{I}$  are known as the independent sets of the matroid and satisfy the following axioms:

- $\emptyset \in \mathcal{I}$ ;
- if  $S \in \mathcal{I}$ ,  $T \subseteq S$ , then  $T \in \mathcal{I}$ ;
- if  $S, T \in \mathcal{I}$  and  $|S| > |T|$ , then there is an element  $i \in S \setminus T$  such that  $T \cup \{i\} \in \mathcal{I}$ .

The first axiom ensures that  $\mathcal{I}$  is non-empty. The second shows that  $\mathcal{I}$  is downward closed, and the third implies that the cardinality of maximal independent sets are the same. This maximum cardinality is known as the *rank* of the matroid  $\mathcal{M}$ . The set of *bases*  $\mathcal{B} = \mathcal{B}(\mathcal{M})$  is the collection of independent sets of maximum cardinality. The rank also extends as a function to subsets of  $U$ . For  $S \subseteq U$ ,  $\text{rk}(S)$  is defined as the size of the maximum independent sets contained in  $S$ .

For a matroid  $\mathcal{M} = (U, \mathcal{I})$ , its *dual matroid*  $M^* = (U, \mathcal{I}^*)$  has the same ground set  $U$  with the collection of independent sets  $\mathcal{I}^* := \{S \subseteq U \mid \exists B \in \mathcal{B}(\mathcal{M}), B \subseteq U \setminus S\}$ . By definition, every base  $B^*$  of  $M^*$  is the complement of the base  $B = U \setminus B^*$  of  $\mathcal{M}$  and vice versa.

### 2.2 Strongly log-concave polynomial

Let  $f \in \mathbb{R}[x_1, x_2, \dots, x_n]$  be a polynomial with non-negative coefficients.  $f$  is called

- *r-homogeneous* if the degree of every monomial in  $f$  is  $r$ ;
- *multiaffine* if every variable appears with degree no more than 1;
- *log-concave over the first orthant* (or *log-concave* for short) if  $\log f$  is concave over  $\mathbb{R}_{>0}^n$ , i.e., for  $x, y \in \mathbb{R}_{>0}^n$  and  $\lambda \in (0, 1)$ ,

$$f(\lambda x + (1 - \lambda)y) \geq f(x)^\lambda f(y)^{1-\lambda};$$

- *strongly log-concave* if  $f$  is either vanishes or log-concave after taking any sequence of partial derivatives.

The notion of strong log-concavity is introduced by Gurvitz [14, 13]. For a homogeneous polynomial, it turns out to be equivalent to related notions of *complete log-concavity* by Anari, Oveis Gharan, and Vinzant [4] and *Lorentzian* by Brändén and Huh [6]. See [6, Theorem 2.30]. For simplicity, we will not define the latter two.

A well known fact is that affine transform  $T : \mathbb{R}^m \rightarrow \mathbb{R}^n$  (i.e.,  $T(\mathbf{y}) = A\mathbf{y} + \mathbf{b}$  for some  $A \in \mathbb{R}^{n \times m}$  and  $\mathbf{b} \in \mathbb{R}^n$ ) preserves log-concavity.

► **Lemma 6** ([4, Lemma 2.1]). *If  $f \in \mathbb{R}[x_1, \dots, x_n]$  is log-concave and  $T : \mathbb{R}^m \rightarrow \mathbb{R}^n$  is an affine transformation such that  $T(\mathbb{R}_{>0}^m) \subseteq \mathbb{R}_{>0}^n$ , then  $f(T(y_1, \dots, y_m)) \in \mathbb{R}[y_1, \dots, y_m]$  is log-concave.*

As observed in [3], for a multiaffine polynomial  $f$ , its partial derivative is given by

$$\partial_1 f = \lim_{c \rightarrow \infty} \frac{f(c, x_2, \dots, x_n)}{c},$$

which means the derivatives of a multiaffine log-concave polynomial are limits of log-concave polynomials, which are also log-concave by definition. It implies that a multiaffine log-concave polynomial is automatically strongly log-concave.

► **Fact 7** ([3]). *If polynomial  $f$  is multiaffine and log-concave, then  $f$  is strongly log-concave.*

We note that most polynomials we consider are multiaffine, which means that log-concavity and strong log-concavity are equivalent within the scope of this work.

## 2.3 Polynomial and distribution

Let  $\mu$  be a distribution over  $2^{[n]}$ . The generating function of  $\mu$  is given by

$$g_\mu = \sum_{S \subseteq [n]} \mu(S) \prod_{i \in S} x_i.$$

It is known that multiaffine polynomials are closely related to generating polynomials of distribution. Let  $f$  be a multiaffine polynomial  $f \in \mathbb{R}[x_1, x_2, \dots, x_n]$  with non-negative coefficients. If  $f \neq 0$ , there exists a distribution  $\mu$  over  $2^{[n]}$  such that its generating polynomial is identical to  $f$  up to a scaling factor. Hence, we may also say  $f$  is the generating polynomial of  $\mu$ .

Furthermore, if the generating function  $g_\mu$  of  $\mu$  is  $r$ -homogeneous and log-concave, then the support of  $\mu$  must be the set of bases of a matroid [6].

## 2.4 Down-up walk

Let  $\mu$  be a distribution over  $\binom{[n]}{r}$ . Let  $\Omega(\mu) := \{S \subseteq [n] \mid \mu(S) > 0\}$  be the support of  $\mu$ . A classical method for sampling from this homogeneous distribution is the down-up walk, described below.

► **Definition 8.** *For a distribution  $\mu$  over  $\binom{[n]}{r}$ , the down-up walk  $P$  updates a configuration  $S \in \binom{[n]}{r}$  according to the following rule:*

1. *select a subset  $T \subseteq S$  of size  $r - 1$  uniformly at random;*
2. *update  $S$  to  $S'$  by selecting  $S' \supseteq T$  with probability proportional to  $\mu(S')$ .*

When the support of  $\mu$  is the set of bases of a matroid, this walk is also known as the *bases-exchange walk*.

If the down-up walk  $P$  connects  $\Omega(\mu)$ , then  $\mu$  is its unique stationary distribution. Its *mixing time* is defined by

$$t_{\text{mix}}(\varepsilon) := \min \left\{ t \mid \max_{S \in \Omega(\mu)} D_{\text{TV}}(P^t(S, \cdot), \mu) \leq \varepsilon \right\}.$$

The down-up walk mixes rapidly if  $g_\mu$  is (strongly) log-concave [7, 3].

► **Proposition 9** ([3, Theorem 1]). *If  $g_\mu$  is  $r$ -homogeneous and log-concave, the mixing time of the down-up walk can be bounded by  $t_{\text{mix}}(\varepsilon) = O(r \log(r/\varepsilon))$ .*

## 3 Our algorithm

In this section, we prove Theorem 1. Our main tool is the down-up walk in Definition 8. As the uniform distribution over independent sets is not homogeneous, in Section 3.1 we first consider a standard homogenization, namely its polarized version. Then standard results imply that the down-up walk for the polarized homogeneous distribution mixes in time  $O(n \log n)$ . In Section 3.2, we show how to implement the down-up walk with  $\tilde{O}(1)$  cost. With these ingredients, the proof of Theorem 1 is given at the end of this section.



### 3.1 Down-up walk for polarized polynomial

Let  $\mathcal{M} = ([n], \mathcal{I})$  be a rank- $r$  matroid and  $\lambda \in \mathbb{R}_{>0}^n$  be the external fields. Consider

$$g(x_1, \dots, x_n) := \sum_{S \in \mathcal{I}} \prod_{i \in S} x_i.$$

It is straightforward to verify that  $g(\lambda_1 x_1, \dots, \lambda_n x_n)$  is the generating polynomial of  $\mu_{\mathcal{M}, \lambda}$ .

Note that  $g$  is not homogeneous, which means that we may not directly employ the down-up walk to sample from the distribution  $\mu_{\mathcal{M}, \lambda}$ . However, there is a homogeneous variant of  $g$ ,

$$g_h(y, x_1, \dots, x_n) := \sum_{S \in \mathcal{I}} y^{n-|S|} \prod_{i \in S} x_i. \quad (5)$$

As a key step in the proof of Mason's ultra-log-concavity conjecture for independent sets of matroid [1, 6], the following result is proved.

► **Lemma 10** ([1, Theorem 4.1]). *The polynomial  $g_h$  in (5) is strongly log-concave.*

However,  $g_h$  is not multiaffine, which means that it is not a generating polynomial of any distribution. Instead, we consider the polarized version of  $g_h$ .

► **Lemma 11.** *If  $g_h$  in (5) is strongly log-concave, then the following polynomial is also strongly log-concave:*

$$g_p(x_1, \dots, x_n, y_1, \dots, y_n) = \sum_{S \in \mathcal{I}} \frac{e_{n-|S|}(\mathbf{y})}{\binom{n}{|S|}} \prod_{i \in S} x_i, \quad (6)$$

where  $e_k(\mathbf{y}) = \sum_{1 \leq i_1 < i_2 < \dots < i_k \leq n} \prod_{j=1}^k y_{i_j}$  is the  $k$ -th elementary symmetric polynomial.

We remark that Lemma 11 is a special case of [6, Proposition 3.1], and it is more explicitly derived in [18, Section 6.6].

The polarized polynomial in (6) corresponds back to a distribution. Let  $X := \{x_1, \dots, x_n\}$  denote elements in  $\mathcal{M}$  and  $Y := \{y_1, \dots, y_n\}$  denote the auxiliary variables introduced by polarization. Let  $\pi$  be the distribution over subsets of  $X \cup Y$  corresponding to the generating polynomial  $g_p(\lambda_1 x_1, \dots, \lambda_n x_n, y_1, \dots, y_n)$ . Then the support of  $\pi$  is given by

$$\Omega(\pi) = \{A \cup B \mid A \in \mathcal{I}, B \subseteq Y, |A| + |B| = n\}.$$

Furthermore, for every  $S = A \cup B \in \Omega(\pi)$ , it holds that

$$\pi(S) \propto \frac{1}{\binom{n}{|A|}} \prod_{x_i \in A} \lambda_i,$$

and  $\sum_{S: S \cap X = A} \pi(S) = \mu_{\mathcal{M}, \lambda}(A)$ .

Therefore, to sample from  $\mu_{\mathcal{M}, \lambda}$  within a TV distance of  $\varepsilon$ , it suffices to sample  $S \in \Omega(\pi)$  such that  $D_{\text{TV}}(S, \pi) \leq \varepsilon$ , and then return  $S \cap X$  as the result.

Note that  $g_p$  is homogeneous and multiaffine. Moreover, according to Lemma 10 and Lemma 11,  $g_p$  is strongly log-concave. Thus, the polynomial  $g_p(\lambda_1 x_1, \dots, \lambda_n x_n, y_1, \dots, y_n)$  is also log-concave by Lemma 6. Hence, by Proposition 9, we have the following result, which gives a powerful framework to build fast sampling algorithm for independent set of matroid.

► **Lemma 12.** *The down-up walk  $P$  of  $\pi$  mixes in time  $O(n \log(n/\varepsilon))$ .*

■ **Algorithm 1** a step of down-up walk  $P$  on  $\pi$ .

- 1 select a subset  $T \subseteq S$  of size  $n - 1$  uniformly at random;
- 2 update  $S$  to  $S'$  by selecting random  $S' \supseteq T$  according to the following law:

$$\Pr[S'] \propto \mathbf{1}[S' \cap X \in \mathcal{I}] \times \begin{cases} \frac{1}{\binom{n}{|T \cap X|}}, & S' \setminus T \in Y; \\ \frac{\lambda_i}{\binom{n}{|T \cap X|+1}}, & S' \setminus T = \{x_i\} \text{ for some } x_i \in X. \end{cases} \quad (7)$$

We note that the mixing time  $O(n \log(n/\varepsilon))$  does not readily imply a sampler which runs in  $O(n \log(n/\varepsilon))$  time, as it may take  $\omega(1)$  time to implement a single transition step of  $P$ . According to Definition 8, in each step,  $P$  updates a state  $S \in \Omega(\pi)$  as in Algorithm 1.

The main obstacle is to implement the second step in Algorithm 1 efficiently. A naïve approach checks whether  $(\{x_i\} \cup T) \cap X \in \mathcal{I}$  for each  $x_i \in X \setminus T$  by calling the independence oracle  $\mathcal{O}_I$ , and then generates a random sample from all “feasible”  $x_i$  and together with all of  $y_i \in Y \setminus T$  according to the desired distribution in (7). In the worst case, this gives an  $O(n)$  overhead and the running time of the sampling algorithm becomes  $O(n^2 \log n)$ .

### 3.2 A fast implementation of the down-up walk

Our main contribution is an efficient implementation of the down-up walk  $P$  on  $\pi$ , where each step of  $P$  takes constant time in expectation given an independence oracle  $\mathcal{O}_I$ . In fact, the implementation task is yet another sampling problem from the distribution in (7), and we do so by rejection sampling, described in Algorithm 2.

■ **Algorithm 2** implementation for the second step of the down-up walk  $P$ .

---

**input** : a subset  $T \subseteq X \cup Y$  of size  $n - 1$  such that  $T \cap X \in \mathcal{I}$   
**output** : a random configuration  $S$  according to the distribution defined in (7)

- 1 **while** *true* **do**
- 2     propose an element  $e \in (X \cup Y) \setminus T$  according to the following distribution  $\nu$ :
 
$$\forall e \in (X \cup Y) \setminus T, \quad \nu(e) \propto \begin{cases} \lambda_i, & e = x_i \in X \setminus T; \\ \frac{n - |T \cap X|}{1 + |T \cap X|}, & e \in Y \setminus T. \end{cases} \quad (8)$$
- 3     **if**  $(T \cup \{e\}) \cap X \in \mathcal{I}$  **then**
- 4         **return**  $S = T \cup \{e\}$ ;

---

The correctness of Algorithm 2 is straightforward.

► **Fact 13.** *The state  $S$  produced by Algorithm 2 follows the distribution defined in (7).*

In terms of efficiency, the **while** loop in Algorithm 2 is anticipated to execute for a constant number of rounds in expectation. This is because the rejection probability is upper bounded by a constant, as shown by the next lemma.

► **Lemma 14.** *It holds that  $\Pr_{e \sim \nu}[(T \cup \{e\}) \cap X \notin \mathcal{I}] \leq \frac{\lambda_{\max}}{1 + \lambda_{\max}}$ , where  $\lambda_{\max} = \max_{i \in [n]} \lambda_i$ .*

**Proof.** Suppose  $|T \cap X| = k$ . Note that if  $e \in Y \setminus T$ , then  $(T \cup \{e\}) \cap X \in \mathcal{I}$ . This means

$$\begin{aligned} \Pr_{e \sim \nu} [(T \cup \{e\}) \cap X \notin \mathcal{I}] &\leq \sum_{x_i \in X \setminus T} \frac{\lambda_i}{\sum_{x_i \in X \setminus T} \lambda_i + \sum_{y \in Y \setminus T} \frac{n-k}{1+k}} \\ &= \frac{\sum_{x_i \in X \setminus T} \lambda_i}{\sum_{x_i \in X \setminus T} (1 + \lambda_i)} \leq \frac{\lambda_{\max}}{1 + \lambda_{\max}}, \end{aligned}$$

where the equality is due to  $|Y \cap T| + k = n - 1$ .  $\blacktriangleleft$

Now, we are ready to prove Theorem 1.

**Proof of Theorem 1.** Our algorithm is just running Algorithm 1 for  $O(n \log(n/\varepsilon))$  steps and then output  $S \cap X$ . Line 2 of Algorithm 1 is implemented by Algorithm 2, to get a random state  $S$ . By Lemma 12 and Fact 13, it holds that  $D_{TV}(S \cap X, \mu) \leq \varepsilon$ .

To implement (8), we maintain two balanced binary search trees  $T_X$  and  $T_Y$  that keep track of the weight of each node and the sum of weights in each subtree. The first tree  $T_X$  maintains elements  $x_i \in X \setminus T$  each assigned with weight  $\lambda_i$ , and the second tree  $T_Y$  maintains elements  $y_i \in Y \setminus T$  with weight 1.

To produce an  $e \sim \nu$ , first choose tree  $T_Z \in \{T_X, T_Y\}$  randomly according to the law:

$$T_Z = \begin{cases} T_X & \text{with prob. } \propto \sum_{x_i \in X \setminus T} \lambda_i \\ T_Y & \text{with prob. } \propto n - |T \cap X| \end{cases},$$

where we note that  $\sum_{x_i \in X \setminus T} \lambda_i$ , and  $|T \cap X|$  could be obtained by a constant number of queries via the binary search trees. To sample an element  $e \in T_Z$  according to the weights of each element, we may consider a binary search algorithm on  $T_Z$  that runs in  $O(\log n)$  time. We initialize a variable  $e$  with the root of  $T_Z$ , and then repeat the following procedure:

1. Let  $L$  be the sum of weights in the left subtrees of  $e$ ,  $R$  be the sum of weights in the right subtrees, and  $W$  be the weights of  $e$ ;
2. Sample a real number  $0 < x < L + R + W$  uniformly at random. If  $x < W$ , return  $e$ ; else if  $x < W + L$ , update  $e$  to the left child of  $e$ ; otherwise, update  $e$  to the right child of  $e$ .

Finally, by Lemma 14, the rejection sampling procedure in Algorithm 2 runs within  $O(1 + \lambda_{\max})$  rounds in expectation. Also note that in each round of the rejection sampling, we need  $t_{\mathcal{O}_I}$  time to query the independence oracle. Together, the algorithm runs in

$$O((1 + \lambda_{\max})n \log(n/\varepsilon)(\log n + t_{\mathcal{O}_I}))$$

time in expectation.  $\blacktriangleleft$

#### 4 Random cluster models with $q \leq 1$

Once again, let  $\mathcal{M} = (X, \mathcal{I})$  be a matroid, equipped with a rank function  $\text{rk}(\cdot)$ . Let  $\lambda = \{\lambda_i\}_{i \in [n]}$ , where  $\lambda_i > 0$  is the weight or external field of  $x_i \in X$ . Let  $0 \leq q \leq 1$  be a parameter. Recall the definition of the random cluster model [9] in (4). Note that when  $q = 0$ , the support of  $\pi_{RC,q,\lambda}$  are all subsets of full rank, namely they are the complements of the independent sets of the dual matroid.

Similar to Section 3.1, as the distribution is not homogeneous, we want to polarize it. Let  $Y$  be a set of  $n = |X|$  auxiliary variables. For  $T \subseteq X \cup Y$  such that  $|T| = n$ , the polarized distribution is given by

$$\widehat{\pi}_{RC,q,\lambda}(T) \propto \frac{q^{-\text{rk}(T \cap X)} \prod_{x_i \in T \cap X} \lambda_i}{\binom{n}{|T \cap Y|}}. \quad (9)$$

### 32:10 Near-Linear Time Samplers for Matroid Independent Sets with Applications

Let the right hand side of (9) be  $\text{wt}(T)$ . Note that the marginal distribution of  $\widehat{\pi}_{RC,q,\lambda}$  on  $X$  is the same as  $\pi_{RC,q,\lambda}$ .

Once homogenized, we may consider the up-down walk for  $\widehat{\pi}_{RC,q,\lambda}$ . For the up step, we uniformly add an element from  $(X \cup Y) \setminus T$ . For the down step, suppose the current set is  $T$  such that  $|T| = n + 1$ . We want to remove an element  $e \in T$  with probability proportional to  $\text{wt}(T \setminus \{e\})$ . Namely, the transition probability  $p(e)$  satisfies

$$p(e) \propto \begin{cases} \frac{q^{-\text{rk}(T \cap X)} \prod_{x_i \in T \cap X} \lambda_i}{\binom{n}{|T \cap Y| - 1}} & \text{if } e \in T \cap Y; \\ \frac{q^{-\text{rk}(T \cap X)} \prod_{x_i \in T \cap X} \lambda_i}{\lambda_j \binom{n}{|T \cap Y|}} & \text{if } e = x_j \in T \cap X \text{ and } \text{rk}(T \cap X \setminus e) = \text{rk}(T \cap X); \\ \frac{q^{-\text{rk}(T \cap X) + 1} \prod_{x_i \in T \cap X} \lambda_i}{\lambda_j \binom{n}{|T \cap Y|}} & \text{if } e = x_j \in T \cap X \text{ and } \text{rk}(T \cap X \setminus e) = \text{rk}(T \cap X) - 1. \end{cases} \quad (10)$$

We may further normalize (10) to get

$$p(e) \propto \begin{cases} \frac{n - |T \cap Y| + 1}{|T \cap Y|} = \frac{|T \cap X|}{|T \cap Y|} & \text{if } e \in T \cap Y; \\ \lambda_j^{-1} & \text{if } e = x_j \in T \cap X \text{ and } \text{rk}(T \cap X \setminus e) = \text{rk}(T \cap X); \\ q \lambda_j^{-1} & \text{if } e = x_j \in T \cap X \text{ and } \text{rk}(T \cap X \setminus e) = \text{rk}(T \cap X) - 1. \end{cases} \quad (11)$$

To implement (11), we may first propose  $e \sim \nu$  where

$$\nu(e) \propto \begin{cases} \frac{|T \cap X|}{|T \cap Y|} & \text{if } e \in T \cap Y; \\ \lambda_j^{-1} & \text{if } e = x_j \in T \cap X, \end{cases}$$

and then reject  $e \in T \cap X$  with probability  $1 - q$  if  $\text{rk}(T \cap X \setminus \{e\}) = \text{rk}(T \cap X) - 1$ . Keep doing this until we accept. To see the efficiency of this implementation. Let  $\mathcal{E}$  be the event that rejection happens. Then,

$$\begin{aligned} \Pr[\mathcal{E}] &\leq \sum_{x_j \in T \cap X} \frac{(1 - q) \lambda_j^{-1}}{\sum_{x_j \in T \cap X} \lambda_j^{-1} + \sum_{e \in T \cap Y} \frac{|T \cap X|}{|T \cap Y|}} \\ &= (1 - q) \frac{\sum_{x_j \in T \cap X} \lambda_j^{-1}}{\sum_{x_j \in T \cap X} (\lambda_j^{-1} + 1)} \leq \frac{1 - q}{1 + \lambda_{\min}}, \end{aligned}$$

where  $\lambda_{\min} := \min_{i \in [n]} \lambda_i$ . Thus, in expectation, we will successfully make a transition after  $O\left(\frac{\lambda_{\min} + 1}{\lambda_{\min} + q}\right) = O(1 + \lambda_{\min}^{-1})$  steps.

One more issue with the implementation is that we need to check the rank of  $T \cap X$  and  $T \cap X \setminus \{e\}$ . This is why we need a rank oracle instead of the independence oracle.

The only missing ingredient to get Theorem 4 is the mixing time of the chain. To this end, we claim that the up-down walk here is just the down-up walk of the corresponding random cluster model on the dual matroid. As the homogenized generating polynomial for random cluster models with  $q \leq 1$  is shown to be strongly log-concave by Brändén and Huh [5], Proposition 9 implies that the mixing time of the up-down walk is  $O(n \log(n/\epsilon))$  as well.

Finally we verify the claim. Let  $\overline{\mathcal{M}} = (X, \overline{\mathcal{I}})$  be the dual matroid of  $\mathcal{M}$ . Consider the random cluster model on  $\overline{\mathcal{M}}$  with external fields  $q/\lambda$ , where  $\lambda = \{\lambda_i\}$  is the external field vector of the original model:

$$\forall U \subseteq X, \quad \pi_{\overline{\mathcal{M}}, q, q/\lambda}(U) \propto q^{-\text{rk}_{\overline{\mathcal{M}}}(U)} \prod_{x_i \in U} \left(\frac{q}{\lambda_i}\right).$$

This is equivalent to having the field  $1/\lambda$ , but the extra  $q$  simplifies some calculation next.

The rank function of the dual matroid satisfies:

$$\text{rk}_{\overline{\mathcal{M}}}(U) = |U| + \text{rk}_{\mathcal{M}}(X \setminus U) - \text{rk}_{\mathcal{M}}(X).$$

As  $\text{rk}_{\mathcal{M}}(X)$  is a constant, we have

$$\forall U \subseteq X, \quad \pi_{\overline{\mathcal{M}},q,q/\lambda}(U) \propto q^{-\text{rk}_{\mathcal{M}}(X \setminus U)} \prod_{x_i \in X \setminus U} \lambda_i.$$

Similarly, the polarized version of  $\pi_{\overline{\mathcal{M}},q,q/\lambda}$  satisfies

$$\forall R \in \binom{X \cup Y}{n}, \quad \widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}(R) \propto \frac{q^{-\text{rk}_{\mathcal{M}}(X \setminus R)} \prod_{x_i \in X \setminus R} \lambda_i}{\binom{n}{|R \cap Y|}}.$$

Comparing the above with (9), we have that

$$\widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}(R) = \widehat{\pi}_{\mathcal{M},q,\lambda}((X \cup Y) \setminus R). \quad (12)$$

The down-up walk (with  $R$  being the current state) on  $\widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}$  is:

- (down) remove an element from  $R$  uniformly at random;
  - (up) add an element  $e \in (X \cup Y) \setminus R$  to  $R$  with probability  $\propto \widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}(R \cup \{e\})$ .
- As  $T = (X \cup Y) \setminus R$ , we can rephrase this down-up walk as an up-down walk on the random cluster over  $\mathcal{M}$ :

- (up) add an element from  $(X \cup Y) \setminus T$  to  $T$  uniformly at random;
- (down) remove an element  $e \in T$  from  $T$  with probability  $\propto \widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}(R \cup \{e\}) = \widehat{\pi}_{\mathcal{M},q,\lambda}(T \setminus \{e\})$ , where the equality is by (12).

Thus, we conclude that the down-up walk on  $\widehat{\pi}_{\overline{\mathcal{M}},q,q/\lambda}$  is the same as the up-down walk on  $\widehat{\pi}_{\mathcal{M},q,\lambda}$  modulo mapping  $T$  to  $(X \cup Y) \setminus T$ . This verifies the claim and finishes the proof of Theorem 4.

---

## References

- 1 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials III: Mason's ultra-log-concavity conjecture for independent sets of matroids. *arXiv*, abs/1811.01600, 2018. [arXiv:1811.01600](#).
- 2 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 *STOC*, pages 1–12. ACM, 2019.
- 3 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, Cynthia Vinzant, and Thuy-Duong Vuong. Log-concave polynomials IV: approximate exchange, tight mixing times, and near-optimal sampling of forests. In *STOC*, pages 408–420. ACM, 2021. [arXiv:2004.07220v2](#).
- 4 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *FOCS*, pages 35–46. IEEE, 2018. [arXiv:1807.00929v2](#).
- 5 Petter Brändén and June Huh. Hodge-Riemann relations for Potts model partition functions. *arXiv*, abs/1811.01696, 2018. [arXiv:1811.01696](#).
- 6 Petter Brändén and June Huh. Lorentzian polynomials. *Ann. of Math. (2)*, 192(3):821–891, 2020. [arXiv:1902.03719v6](#).
- 7 Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-Sobolev inequalities for strongly log-concave distributions. *Ann. Probab.*, 49(1):506–525, 2021.
- 8 Tomás Feder and Milena Mihail. Balanced matroids. In *STOC*, pages 26–38. ACM, 1992.
- 9 Cornelius M. Fortuin and Pieter W. Kasteleyn. On the random-cluster model. I. Introduction and relation to other models. *Physica*, 57:536–564, 1972.

- 10 Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Structures Algorithms*, 57(2):371–392, 2020.
- 11 Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.
- 12 Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):18:1–18:31, 2019.
- 13 Leonid Gurvits. On multivariate Newton-like inequalities. In *Advances in combinatorial mathematics*, pages 61–78. Springer, Berlin, 2009.
- 14 Leonid Gurvits. A polynomial-time algorithm to approximate the mixed volume within a simply exponential factor. *Discrete Comput. Geom.*, 41(4):533–555, 2009.
- 15 Mark Jerrum. *On the complexity of evaluating multivariate polynomials*. PhD thesis, The University of Edinburgh, 1981.
- 16 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43(2-3):169–188, 1986.
- 17 Vladimir Kolmogorov. A faster approximation algorithm for the gibbs partition function. In *COLT*, volume 75, pages 228–249. PMLR, 2018.
- 18 Giorgos Mousa. *Local-to-Global Functional Inequalities in Simplicial Complexes*. PhD thesis, The University of Edinburgh, 2022.
- 19 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.
- 20 Daniel Štefankovič, Santosh Vempala, and Eric Vigoda. Adaptive simulated annealing: a near-optimal connection between sampling and counting. *J. ACM*, 56(3):Art. 18, 36, 2009.
- 21 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.
- 22 Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *SODA*, pages 1757–1769. SIAM, 2013.

# On the Amortized Complexity of Approximate Counting

Ishaq Aden-Ali   




University of California, Berkeley, CA, USA

Yanjun Han   

New York University, NY, USA

Jelani Nelson   

University of California, Berkeley, CA, USA

Huacheng Yu   

Princeton University, NJ, USA

---

## Abstract

---

Naively storing a counter up to value  $n$  would require  $\Omega(\log n)$  bits of memory. Nelson and Yu [9], following work of Morris [8], showed that if the query answers need only be  $(1 + \epsilon)$ -approximate with probability at least  $1 - \delta$ , then  $O(\log \log n + \log \log(1/\delta) + \log(1/\epsilon))$  bits suffice, and in fact this bound is tight. Morris' original motivation for studying this problem though, as well as modern applications, require not only maintaining one counter, but rather  $k$  counters for  $k$  large. This motivates the following question: for  $k$  large, can  $k$  counters be simultaneously maintained using asymptotically less memory than  $k$  times the cost of an individual counter? That is to say, does this problem benefit from an improved *amortized* space complexity bound?

We answer this question in the negative. Specifically, we prove a lower bound for nearly the full range of parameters showing that, in terms of memory usage, there is no asymptotic benefit possible via amortization when storing multiple counters. Our main proof utilizes a certain notion of “information cost” recently introduced by Braverman, Garg and Woodruff [2] to prove lower bounds for streaming algorithms.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Lower bounds and information complexity

**Keywords and phrases** streaming, approximate counting, information complexity, lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.33

**Category** RANDOM

**Funding** *Ishaq Aden-Ali*: supported by ONR DORECG award N00014-17-1-2127.

*Yanjun Han*: work done while at UC Berkeley, supported by the Berkeley-Simons Research Fellowship and Norbert Wiener postdoctoral fellowship.

*Jelani Nelson*: supported by NSF award CCF-1951384, ONR grant N00014-18-1-2562, and ONR DORECG award N00014-17-1-2127.

**Acknowledgements** We thank Sidhanth Mohanty for very enlightening discussions on unpredictable paths, half Cauchy random variables, and stochastic processes in general that ultimately led to the discovery of the first version of our main lower bound. We also thank Mark Sellke for answering a certain question regarding stochastic processes. Lastly, we thank Greg Valiant for raising the question of the amortized space complexity of approximate counting.

## 1 Introduction

Maintaining a counter subject to increments is one of the most basic data structural problems in computer science. If the counter value stays below  $n$ , then the naive solution of maintaining the counter explicitly consumes  $O(\log n)$  bits of memory. In 1978, Morris devised a new



© Ishaq Aden-Ali, Yanjun Han, Jelani Nelson, and Huacheng Yu;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 33; pp. 33:1–33:17



Leibniz International Proceedings in Informatics

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

*approximate counting* algorithm which could maintain such a dynamic counter using much fewer bits in expectation [8]. His algorithm uses a number of random bits which itself is a random variable, whose expectation is  $O(\log \log n + \log(1/\epsilon) + \log(1/\delta))$ . At query time, this *Morris Counter* returns the correct counter value up to  $(1+\epsilon)$ -multiplicative error with failure probability at most  $\delta$ . Recently, Nelson and Yu [9] showed that a slight alteration to Morris' algorithm improves the dependence on  $\delta$ , consuming only  $O(\log \log n + \log(1/\epsilon) + \log \log(1/\delta))$  bits of memory. They also proved that this bound is asymptotically optimal.

Morris' original motivation for developing his approximate counter in the mid 1970s was, for each of  $26^3$  possible trigrams (sequences of three characters), to count the number of occurrences of that trigram amongst words in some text; these counts were then used by **typo**, an early Unix spellchecker (see more on this history in a survey by Lumbroso [7]). Approximate counters still find use in modern-day applications. For example, Redis is a highly popular in-memory database which is often used as a cache, and one built-in cache replacement policy is Least Frequently Used. To implement LFU, each item stored in cache must have an associated counter, counting the number of queries to that item over some time window. These counters are currently implemented in Redis using a variant of the Morris Counter [6].

Both the original application of Morris as well as modern-day applications of approximate counting, such as that described in Redis, need not store just one counter but many. Even in the 1970s, Morris' computer would have had enough memory to count the number of occurrences of a single trigram, but the difficulty was keeping counts of all  $26^3$  counters in memory simultaneously. Similarly, modern-day computers and mobile devices clearly also contain enough memory to store a single counter for any reasonable counter value, but one may wish to minimize the space per counter in an application where many counters must be stored simultaneously, as in the case of Redis being used to cache a large number of items. Thus even the original motivation of Morris inspires the following question:

*What is the **amortized** space complexity of approximate counting? Specifically, for  $k$  large, is it possible to maintain  $k$  approximate counters using less memory than  $k$  times that of an individual approximate counter?*

Typically such lower bounds are obtained via a direct sum argument in combination with an information complexity lower bound [3]. Specifically, a common way to carry out proving such lower bounds in the streaming context is as follows [1]: first, devise a communication game that you believe is hard (requires a lot of communication) to solve, and which reduces to the streaming problem (i.e., a low-memory streaming algorithm would imply a low-communication protocol). Suppose for illustration that this communication game is one-way, with Alice receiving input  $X$  who sends a single message  $\Pi$  to Bob who holds input  $Y$ . Here we use distributional complexity, in which  $(X, Y)$  is drawn from some distribution  $\mathcal{D}$ , and the goal is to devise a protocol in which Bob computes some  $f(X, Y)$  with success probability at least  $1 - \delta$ . Now, the so-called *external information complexity* of  $\Pi$  is defined as  $I(\Pi; X, Y)$ , which is at most  $|\Pi|$  and thus it suffices to lower bound  $I(\Pi; X, Y)$ . Once one establishes a lower bound on this quantity, then for a sequence  $\{(X_i, Y_i)\}_{i=1}^k$  of  $k$  of independently drawn such inputs, we have the following inequality [3, Theorem 1.5]:  $I(\Pi; X_1, Y_1, X_2, Y_2, \dots, X_k, Y_k) \geq \sum_{i=1}^k I(\Pi; X_i, Y_i)$ . Note though that if we want a lower bound for  $\Pi$  solving the  $k$ -fold problem, then we cannot immediately invoke the single-instance lower bound to bound each summand on the right hand side (since this  $\Pi$  is a protocol solving the  $k$ -fold problem, not a single instance!). One then typically completes the argument by showing how to efficiently embed the single-instance communication game into that of  $k$  parallel instances, to then lift the single-instance lower bound to a lower bound for the  $k$ -fold problem as outlined above.



The trouble with obtaining our desired lower bound for approximate counting via the above paradigm is that the memory lower bound of [9] was not proven via information complexity, and thus it is not clear how to execute the above direct sum argument. Rather, the lower bound proof there followed an approach similar to the proof of the pumping lemma for regular languages [10].

**Our Contribution.** We answer the above question in the negative for nearly the full range of parameters (for technical reasons in our analysis, we require the restriction  $\epsilon < 1/\log \log(1/\delta)$ ). Specifically, for this range of  $\epsilon$  we prove a strong memory lower bound demonstrating that there is no asymptotic benefit from amortization when maintaining multiple approximate counters. Our main approach is to first establish a novel information complexity type lower bound for a single approximate counter, which we then lift to multiple counters using a standard direct sum argument, by constructing an embedding of the single-counter problem into the  $k$ -fold problem.

Below we formally define the problem we solve, then state our main result. The approximate counting problem for a single counter is to maintain a multiplicative approximation to a count  $N$  (initially 0) that undergoes a sequence of increment operations. Specifically, the algorithm should support two operations: `increment()` increments  $N$  by 1, and `query()` returns a value  $\hat{N}$  such that  $\mathbb{P}(|\hat{N} - N| \geq \epsilon N) \leq \delta$ ; that is, at any point in time the data structure should be able to provide a  $(1 + \epsilon)$  multiplicative approximation with probability at least  $1 - \delta$ . Our generalization to approximating  $k$  counters is as follows:

► **Definition 1** ( $(k, \epsilon, \delta)$ -approximate counter). *Let  $N_1, \dots, N_k$  be  $k$  counters all initially set to 0. We say a randomized algorithm  $\mathcal{A}$  is a  $(k, \epsilon, \delta)$ -approximate counter if it supports two operations: `increment( $i$ )` increments  $N_i$  by 1, and `query( $i$ )` returns a value  $\hat{N}_i$  such that*

$$\mathbb{P}\left(|\hat{N}_i - N_i| \geq \epsilon N_i\right) \leq \delta.$$

When  $k = 1$ , we will simply call this an  $(\epsilon, \delta)$ -approximate counter.

The following is then our main theorem, when here and henceforth we use  $n$  in lower bounds to denote an upper bound on the number of `increment` operations:

► **Theorem 2.** *For any  $\delta < c_1$  and  $\epsilon < c_2/\log \log(1/\delta)$ , if  $k < c_3 n$  then after a sequence of at most  $n$  updates any  $(k, \epsilon, \delta)$ -counter must use  $\Omega(k \min\{\log(n/k), \log \log(n/k) + \log(1/\epsilon) + \log \log(1/\delta)\})$  bits of memory in expectation. Else if  $k > c_3 n$ , then a lower bound of  $\Omega(n \log(2k/(c_3 n)))$  holds;  $c_1, c_2, c_3 \geq 0$  are universal constants. Furthermore, for both ranges of  $k$ , these lower bounds are tight up to constant factors.*

## 1.1 Preliminaries and notation

We write  $X \sim \mathcal{D}$  to represent a random variable  $X$  sampled from the probability distribution  $\mathcal{D}$ . If the distribution of  $X$  has not been explicitly defined, we write  $P_X$  to the corresponding probability distribution of  $X$ . For two probability distributions  $P$  and  $Q$  defined on the same domain, we write  $\text{TV}(P, Q) = \int |dP - dQ|$  to be their total variation distance and  $\text{KL}(P||Q) = \int dP \log(dP/dQ)$  their Kullback-Leibler divergence. We frequently make use of Pinsker's inequality  $\text{TV}(P, Q) \leq \sqrt{\text{KL}(P||Q)}/2$ . For random variables  $X$  and  $Y$ , we write  $H(X) = \mathbb{E}_{x \sim P_X}[\log(1/P_X(x))]$  to denote the (Shannon) entropy and  $H(X | Y) = \mathbb{E}_{(x,y) \sim P_{XY}}[\log(1/P_{X|Y}(x | y))]$  to be the conditional (Shannon) entropy. The mutual

### 33:4 On the Amortized Complexity of Approximate Counting

information between two random variables  $X$  and  $Y$  is  $I(X; Y) := H(X) - H(X | Y) = H(Y) - H(Y | X)$ . The conditional mutual information is  $I(X; Y | Z) := H(X | Z) - H(X | Y, Z)$ . We frequently make use of the following inequality:

$$I(X; Y | Z) = \mathbb{E}_{X, Z} [\text{KL}(P_{Y|Z} \| P_{Y|X, Z})].$$

We also use the following well known facts about conditional mutual information [4]:

► **Proposition 3 (Chain rule).** *For random variables  $X_1, X_2, Y, Z$  we have*

$$I(X_1, X_2; Y | Z) = I(X_1; Y | Z, X_2) + I(X_2; Y | Z).$$

► **Proposition 4 (Superadditivity).** *Let  $X_1, \dots, X_n, Y$  and  $Z$  be random variables such that  $X_1, \dots, X_n$  are conditionally independent given  $Z$ . Then*

$$\sum_{i=1}^n I(X_i; Y | Z) \leq I(X_1, \dots, X_n; Y | Z).$$

Lastly,  $c, c_1, c_2, \dots > 0$  represent universal constants that may change from statement to statement.

## 1.2 Proof overview

Now we present an overview of our lower bound proof. It turns out that the terms  $\Omega(k(\log \log(n/k) + \log(1/\epsilon)))$  in the lower bound can be proved using a similar argument to the single counter case, which holds even in the offline setting, and the main challenge is to prove the dependence on  $\delta$ ,  $\Omega(k \log \log(1/\delta))$ . The previous proof, for single counter, of  $\Omega(\log \log(1/\delta))$  is based on a pumping-lemma argument, which crucially uses the fact that all updates are exactly the same, i.e., incrementing the counter. However, this no longer holds with multiple counters – we may increment any one of the  $k$  counters each time, and there are  $k$  different updates we can perform. The previous argument fails to generalize.

As we mentioned in the introduction, we first (re)prove an *information theoretic* lower bound for single counter, then apply the *superadditivity of mutual information* for independent variables to obtain the direct-sum result. For simplicity, in this overview we will focus on the case where  $\delta = 2^{-\Theta(n/k)}$ , and prove an  $\Omega(k \log \log(1/\delta)) = \Omega(k \log(n/k))$  lower bound when  $\epsilon \leq O(1/\log(n/k))$ . This case captures all the main ideas, and generalizing to the full lower bound is straightforward.

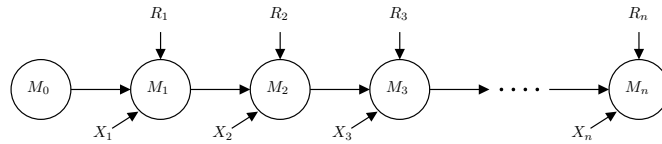
To facilitate the argument, we first slightly reformulate the problem:<sup>1</sup> Consider a stream with  $T$  batches of updates (think  $T = (n/k)^{0.1}$ ), and in each batch  $i$ , the inputs are  $k$  nonnegative integers  $X_{i,c}$  for  $c \in [k]$ , which are the number of increments we apply to each counter in this batch. Since the batch number  $i$  takes only  $O(\log T)$  bits to maintain, we may assume without loss of generality that it is given to the algorithm for free. Clearly if there is an algorithm that approximately maintains  $k$  counters, this reformulation also admits a solution using the same space, provided  $\sum_{i,c} X_{i,c} \leq n$ . After the reformulation, the single counter problem simply has  $T$  nonnegative numbers  $X_i$  as the input stream, and we would like to approximate their sum using small space provided that the sum is at most poly  $T$ , again assuming that  $i$  is given to the algorithm for free.

We will design a hard distribution over the input streams, and analyze the failure probability and measure “information” with respect to this distribution. It is crucial that the notion of information cost we use for streaming algorithms is chosen carefully. The

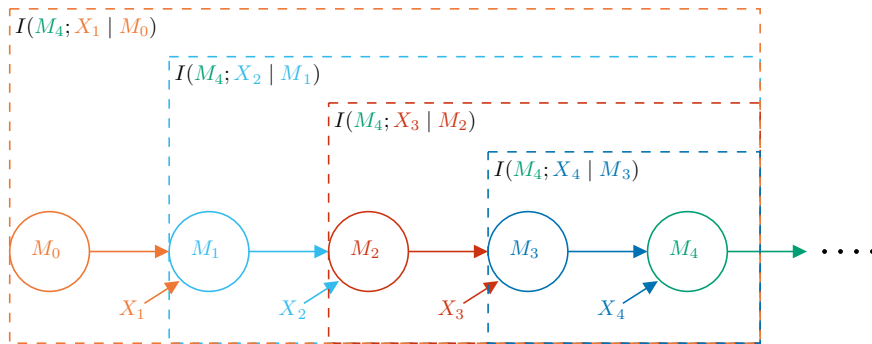
<sup>1</sup> For simplicity, the overview uses slightly different notation and parameters than the actual proof.

information cost defined in [2] turns out to be the right measure for our problem. Fix a streaming algorithm, and let  $M_i$  be its memory state immediately after processing  $X_i$  (see Figure 1). When all numbers  $X_i$  are independent, which will be the case for our distribution, the *information cost* of this algorithm is defined as follows (see Figure 2)

$$\text{IC} := \sum_{i=1}^T \sum_{j=1}^i I(M_i; X_j | M_{j-1}).$$



■ **Figure 1** A depiction of the evolution of the memory state of a randomized streaming algorithm  $\mathcal{A}$ .  $X_1, \dots, X_n$  are the inputs the streaming algorithm receives and  $R_1, \dots, R_n$  are the independent random bits  $\mathcal{A}$  uses as it processes the stream.  $M_0$  is the (possibly random) initial state and  $M_i$  is the state of  $\mathcal{A}$  after processing  $X_i$ . Note that  $M_i$  is a deterministic function of the previous memory state  $M_{i-1}$ , the  $i$ th input  $X_i$ , and the random bits  $R_i$ . When the inputs are random variables, this figure also depicts the dependence structure of the joint distribution of the random variables  $(X_1, \dots, X_n, R_1, \dots, R_n, M_0, M_1, \dots, M_n)$ .



■ **Figure 2** An illustration of a single inner sum of the information cost  $\sum_{j \leq i} I(M_i; X_j | M_{j-1})$  for  $i = 4$ . To simplify the presentation we do not include the random bits that the streaming algorithm uses while processing the stream in this illustration.

As it was shown in [2], this quantity lower bounds  $\sum_{i=1}^T |M_i|$ , i.e.,  $T$  times the memory consumption, and it satisfies the direct-sum property: solving  $k$  independent instances of the problem needs exactly  $k$  times IC. Thus, it suffices to prove an IC lower bound of  $\Omega(T \log T) = \Omega(T \log(n/k))$  for a single approximate counter. Intuitively, such a lower bound means that the algorithm must spend  $\Omega(1)$  bits remembering each bit of the sum (recall that we ensure  $\sum X_i \leq \text{poly } T$ ).

Let us first focus on the lowest bit, i.e., the parity of the sum. Note that one does not have to know the lowest bit in order to return an approximation of the sum. Nevertheless, we will show that the algorithm has to constantly pay attention to this bit in order to output a good approximation with very high probability. To this end, let us consider the distribution where the  $X_i$ 's are independent and uniform in  $\{0, 1\}$ . Let us focus on the terms in IC with  $j = i$ , i.e., those of the form  $I(M_i; X_i | M_{i-1})$ . Intuitively, under this distribution, this term represents how much attention the algorithm is paying to the evolution of the parity at batch  $i$ . This is because  $X_i$  is simply the difference between the parities of the first  $i - 1$  and first  $i$  inputs.

### 33:6 On the Amortized Complexity of Approximate Counting

Suppose that  $I(M_i; X_i \mid M_{i-1}) \ll 1$  for a constant fraction of  $i$ . Then, we can show that the algorithm will make an error of  $\Omega(T)$  with at least  $\delta$  probability. Roughly speaking, this is because for each such  $i$ , conditioned on  $M_i$  and  $M_{i-1}$ , the distribution of  $X_i$  is still close to uniform (as  $X_i$  is uniform conditioned on  $M_{i-1}$ ). Therefore, if we condition on all  $M_0, M_1, \dots, M_T$ , most  $X_i$  can still take both values 0 or 1 with constant probability, and all  $X_i$  are still independent by the Markov property of the algorithm. In particular, by setting all these  $X_i$ 's to 0 or setting all to 1, we reach the same final memory state  $M_T$ , but in the two cases, the total sum differs by  $\Omega(T)$ . Since both happen with probability  $2^{-O(T)} \gg \delta$  given the final memory state  $M_T$ , the algorithm must make an error of at least  $\Omega(T)$  with probability at least  $\delta$ .

We can extend this argument to any specific bit of the sum by considering a stream with  $\Theta(T/B^l)$  independent random increments that are uniform in  $\{0, B^l\}$  for some constant  $B$  and  $l \in [L]$ , where  $L = \log_B T$ . Our final hard distribution interleaves  $L$  such streams, which we call the scales. For each scale  $l \in [L]$ , we evenly spread the  $\Theta(T/B^l)$  random increments in the whole stream with a gap of  $\Theta(B^l)$  batches. Note that now the sum of all inputs is always at most  $O(T \log T)$ . For the sum in the definition of IC, we will only focus on  $L$  terms for each  $i$ :  $I(M_i; X_{\lfloor i \rfloor_l} \mid M_{\lfloor i \rfloor_l - 1})$ , where  $X_{\lfloor i \rfloor_l}$  is the closest scale  $l$  input before  $M_i$ . If  $\text{IC} \ll T \cdot L = O(T \log T)$ , then there must exist some scale  $l^*$  such that  $I(M_i; X_{\lfloor i \rfloor_{l^*}} \mid M_{\lfloor i \rfloor_{l^*} - 1}) \ll 1$  for most  $i$ . Then we apply the above argument, and show that conditioned on the memory states right before each scale- $l^*$  input, most scale- $l^*$  inputs can still take values 0 or  $B^{l^*}$  with constant probability, and the scale- $l^*$  inputs are still independent. We further observe that for most of them, the inputs between  $X_{\lfloor i \rfloor_{l^*}}$  and  $M_i$  are coming from the lower scales  $l < l^*$ . The standard deviation of their sum is much smaller than  $B^{l^*}$ , and we can show that they do not affect the sum by too much as we alter  $X_{\lfloor i \rfloor_{l^*}}$ . Thus, by setting these scale- $l^*$  inputs to 0 or  $B^{l^*}$ , the entire sum will again differ by  $\Omega(T)$ , but they lead to the same final memory state, i.e., the algorithm does not distinguish between the two cases. Since the sum is  $O(T \log T)$ , such a difference is more than  $\epsilon$  times the sum when  $\epsilon < O(1/\log T)$ . A more careful analysis shows that this happens with probability at least  $2^{-O(T)} > \delta$ , leading to a contradiction.

#### Discussion on the choice of the hard distribution

We note here that the independent uniform  $\{0, 1\}$  distribution, by itself, is *not* hard for the information cost. One solution with low information cost is to divide  $X_i$ 's into blocks of size  $S = \Theta(\epsilon^{-2} \log T)$ , and maintain the exact sum within the current block. If *all* blocks have sums  $(1 \pm \epsilon)S/2$ , then we simply remember this fact and use  $T/2$  as the final output. Otherwise, the algorithm finds a block whose sum is not in this range, then it maintains the exact sum for all future blocks. Since all previous blocks have sums  $(1 \pm \epsilon)S/2$ , in particular,  $S/2$  is a  $(1 + \epsilon)$ -approximation for them, the algorithm can also return a  $(1 + \epsilon)$ -approximation of the whole sum. Now note that this case only happens with  $1/\text{poly}(T)$  probability, and the total information cost is at most  $T$  times the expected memory usage. The expected memory is at most  $O(\log(1/\epsilon) + \log \log T)$  bits for maintaining the sum in the current block, plus  $O(T^{-\Theta(1)} \cdot \log T)$  bits in expectation for maintaining the entire exact sum. When  $\epsilon = \Theta(1/\log T)$ , the information cost is only  $O(T \log \log T) \ll T \log T$ .

The above strategy works since the sum of block is concentrated around the expectation, hence, we need extra space to maintain the exact sum only with very small probability. One can also show that the above strategy also applies to any i.i.d. distributions with some concentration.<sup>2</sup> Our hard distribution is inspired by the *discrete half-Cauchy distribution*,

<sup>2</sup> For example, it suffices to have finite  $\mathbb{E}[|X|^{1.01}]$ .

which has probability  $\Theta(1/(x+1)^2)$  at integers  $x \geq 0$ . This distribution does not have an expectation, hence, there is no concentration for blocks of any size. We also have a more involved proof of our main result that uses this distribution instead; the main property that proof relies on is that for every  $W$  random variables, roughly one of them takes value  $\Theta(W)$ . The hard distribution we actually use in this paper is a bounded distribution that can be viewed as extracting this useful property of the half-Cauchy distribution, but which can be analyzed more simply.

## 2 Lower Bounds

In this section we will prove our lower bound for the space complexity of  $(k, \epsilon, \delta)$ -approximate counting. We split the proof into two parts. We first focus on proving the difficult part of the lower bound that depends on the failure probability  $\delta$ . To do so, we derive a lower bound for the space complexity of  $(\epsilon, \delta)$ -approximate counting by appealing to an information theoretic argument. By using a good definition of information cost together with an appropriately chosen hard distribution, we can prove that any accurate algorithm remember a lot of information about many different parts of the stream, i.e. it incurs a high information cost. This immediately gives us a lower bound on the memory size. We then use this result to prove a space lowerbound for  $(k, \epsilon, \delta)$ -approximate counting via a direct sum argument. We conclude the section by proving the portion of the lower bound that depends on the approximation error  $\epsilon$  and the total sum of all counters  $n$  by generalizing the argument used in the single counter case.

### 2.1 Information lower bound for a single counter

Recall that for a randomized streaming algorithm  $\mathcal{A}$  we define  $M_i$  to be the state of  $\mathcal{A}$  after processing the  $i$ th input  $X_i$  together with some additional independent random bits  $R_i$ , i.e.  $M_i$  is a deterministic function of  $M_{i-1}$ ,  $X_i$  and  $R_i$  (equivalently, a deterministic function of  $X_{\leq i}$  and  $R_{\leq i}$ ). The following is a notion of information cost for streaming algorithms originally defined by Braverman, Garg, and Woodruff [2].

► **Definition 5.** *Let  $\mathcal{A}$  be a randomized streaming algorithm. Given a distribution  $\mathcal{D}$  over input sequences of length  $s$ , we define the information cost of algorithm  $\mathcal{A}$  on input  $(X_1, \dots, X_s) \sim \mathcal{D}$  to be*

$$\text{IC}(\mathcal{A}, \mathcal{D}) := \sum_{i=1}^s \sum_{j=1}^i I(M_i; X_j \mid M_{j-1}).$$

The above definition of the information cost is motivated by the following chain of inequalities:

$$\begin{aligned} \mathbb{E}|M_i| &\geq H(M_i) \text{ (source coding theorem}^3\text{)} \\ &\geq I(M_i; X_{\leq i}, M_{<i}) \\ &= \sum_{j=1}^i I(M_i; X_j, M_{j-1} \mid X_{<j}, M_{<j-1}) \\ &= \sum_{j=1}^i I(M_i; M_{j-1} \mid X_{<j}, M_{<j-1}) + I(M_i; X_j \mid X_{<j}, M_{<j}) \\ &= \sum_{j=1}^i I(M_i; X_j \mid X_{<j}, M_{<j}) \\ &= \sum_{j=1}^i I(M_i; X_j \mid M_{j-1}). \end{aligned}$$

### 33:8 On the Amortized Complexity of Approximate Counting

This implies that  $\text{IC}(\mathcal{A}; \mathcal{D}) \leq \sum_{i=1}^s \mathbb{E} |M_i|$ . The main technical part of this paper is proving the following lower bound for a single counter using this notion of information cost.

► **Lemma 6.** *Let  $\mathcal{A}$  be a  $(\epsilon, \delta)$ -approximate counter with parameters  $\delta \in (0, c_1)$  and  $\epsilon \in (0, \frac{c_2}{\log \log(1/\delta)})$  that uses  $|M|$  bits of space. There is a distribution  $\mathcal{D}$  over inputs such that the information cost of  $\mathcal{A}$  on  $(X_1, \dots, X_n) \sim \mathcal{D}$  satisfies*

$$\text{IC}(\mathcal{A}; \mathcal{D}) = \Omega(n \log \log(1/\delta)),$$

where  $\mathbb{P}[\sum_{i=1}^n X_i \leq n] = 1$  and  $n \geq c_3 \log^{c_4}(1/\delta)$ . This implies the space lower bound

$$\frac{1}{n} \sum_{i=1}^n \mathbb{E} |M_i| = \Omega(\log \log(1/\delta)) = \Omega(\min\{\log n, \log \log(1/\delta)\}).$$

**Proof.** We construct the distribution  $\mathcal{D}$  as follows: let  $B \in \mathbb{N}$  be a large integer constant to be specified later, and  $T$  is the largest power of  $B$  such that  $T < \log_{32}(1/\delta)$ . When  $\delta < c_1$  where  $c_1 = c_1(B)$  is a sufficiently small constant, we have  $T \geq B$  and so  $L := \log_B T \in \mathbb{N}$ . The distribution  $\mathcal{D}$  is an interleaving of  $L$  distributions  $\mathcal{D}_\ell$  on  $L$  different scales, where for each scale  $\ell \in [L]$ , the distribution  $\mathcal{D}_\ell$  is a product distribution  $\prod_{i=1}^T \mathcal{D}_{\ell,i}$ :

$$\text{under } \mathcal{D}_{\ell,i}, \quad Y_{\ell,i} \begin{cases} \sim \text{Unif}(\{0, B^\ell\}) & \text{if } B^\ell \text{ divides } (i-1), \\ = 0 & \text{otherwise.} \end{cases}$$

For notational simplicity, we also denote the non-zero entries of  $Y_{\ell,i}$  by  $Z_{\ell,j} = Y_{\ell, B^\ell(j-1)+1}$  for  $j \in [T/B^\ell]$ . The stream under distribution  $\mathcal{D}$  is then generated by interleaving the  $Y_{\ell,i}$  terms to form the sequence  $X^n = (Y_{1,1}, \dots, Y_{L,1}, Y_{1,2}, \dots, Y_{L,2}, \dots, Y_{L,T})$ , where  $n := TL$  is the length of the sequence. The total value of the counter under this stream is at most

$$\sum_{\ell=1}^L B^\ell \cdot \frac{T}{B^\ell} = n.$$

Finally, note that  $n = TL \geq c_3 \log^{c_4}(1/\delta)$  for an appropriate choice of constants  $c_3, c_4 \geq 0$ .

Let  $\mathcal{A}$  be an  $(\epsilon, \delta)$ -approximate counter, and assume towards contradiction that  $\text{IC}(\mathcal{A}; \mathcal{D}) < \alpha n L$  where  $\alpha$  is a small constant. For any index  $i \in [n]$  in the stream, let  $[i]_\ell := B^\ell L \lfloor (i-1)/(B^\ell L) \rfloor + \ell$  be the index of the closest  $Z_{\ell,j}$  to the left of  $X_i$  in the stream, i.e.  $X_{[i]_\ell} = Z_{\ell,j}$ . Similarly, for  $j \in [T/B^\ell]$ , we define  $\lceil j \rceil_\ell := B^\ell L(j-1) + \ell$  to be the index of the  $j$ th non-zero entry of scale  $\ell$  in the stream, i.e.  $X_{\lceil j \rceil_\ell} = Z_{\ell,j}$ . The definition of the information cost tells us that

$$\begin{aligned} \alpha n L &> \text{IC}(\mathcal{A}; \mathcal{D}) \\ &= \sum_{i=1}^n \sum_{j \leq i} I(M_i; X_j \mid M_{j-1}) \\ &\geq \sum_{i=1}^n \sum_{\ell=1}^L I(M_i; X_{[i]_\ell} \mid M_{\lceil [i]_\ell / B^\ell \rceil_\ell}) \end{aligned}$$

<sup>3</sup> The source coding theorem holds for any prefix code. In general we may lose a factor of 2 in this inequality: we have both  $\mathbb{E}[|M_i|] \geq H(M_i \mid |M_i|)$  and  $\mathbb{E}[|M_i|] = \sum_{n \geq 1} n p_n = \sum_{n \geq 1} p_n \log(1/2^{-n}) = \sum_{n \geq 1} p_n \log(p_n/2^{-n}) + H(|M_i|) \geq H(|M_i|)$  for  $p_n := \mathbb{P}(|M_i| = n)$ ; consequently  $2\mathbb{E}[|M_i|] \geq H(M_i \mid |M_i|) + H(|M_i|) \geq H(M_i)$ .

$$\begin{aligned}
&= \sum_{\ell=1}^L \sum_{j=1}^{T/B^\ell} \sum_{i=[j]_\ell}^{[j+1]_\ell-1} I(M_i; X_{[j]_\ell} \mid M_{[j]_\ell-1}) \\
&\geq \sum_{\ell=1}^L \sum_{j=1}^{T/B^\ell} B^\ell L \cdot I(M_{[j+1]_\ell-1}; X_{[j]_\ell} \mid M_{[j]_\ell-1}) \\
&\geq \min_{\ell \in [L]} B^\ell L^2 \sum_{j=1}^{T/B^\ell} I(M_{[j+1]_\ell-1}; X_{[j]_\ell} \mid M_{[j]_\ell-1}) \\
&= \min_{\ell \in [L]} B^\ell L^2 \sum_{j=1}^{T/B^\ell} I(M_{[j+1]_\ell-1}; Z_{\ell,j} \mid M_{[j]_\ell-1})
\end{aligned}$$

where the second last inequality is due to the data processing inequality. By Markov's inequality, there exist  $\ell_0 \in [L]$  and  $J_0 \subseteq [T/B^{\ell_0}]$  with  $|J_0| \geq T/(2B^{\ell_0})$  and

$$I(M_{[j+1]_{\ell_0-1}}; Z_{\ell_0,j} \mid M_{[j]_{\ell_0-1}}) \leq 2\alpha, \quad \forall j \in J_0.$$

For ease of presentation, we abuse notation and write the above inequality as  $I(M_j; Z_j \mid M_{j-1}) \leq 2\alpha$  for  $j \in J_0$ . We shall also keep in mind that the stream between  $M_{j-1}$  and  $M_j$  contains  $(Z_j^<, Z_j, Z_j^>)$ , where  $Z_j^<$  is the collection of all non-zero inputs  $\{Z_{\ell',j'}\}$  inside this window with a lower scale  $\ell' < \ell$ , and  $Z_j^>$  is the counterpart with a higher scale  $\ell' > \ell$  (see Figure 3).

Next we define several good events for the sake of analysis. The first good event  $E_{j,1}$  characterizes the behavior of the contribution of  $Z_j^<$  and is formally defined as

$$\mathbb{1}(E_{j,1}) := \mathbb{1} \left( \left| \text{sum}(Z_j^<) - \mathbb{E}[\text{sum}(Z_j^<) \mid M_{j-1}, M_j] \right| \leq \frac{B^{\ell_0}}{4} \right), \quad j \in J_0.$$

By Chebyshev's inequality, it is clear that

$$\begin{aligned}
\mathbb{E}_{M_{j-1}, M_j} [\mathbb{P}(E_{j,1}^c \mid M_{j-1}, M_j)] &\leq \mathbb{E}_{M_{j-1}, M_j} \left[ \frac{\text{Var}(\text{sum}(Z_j^<) \mid M_{j-1}, M_j)}{(B^{\ell_0}/4)^2} \right] \\
&\leq \frac{\text{Var}(\text{sum}(Z_j^<))}{(B^{\ell_0}/4)^2} \\
&= \frac{16}{B^{2\ell_0}} \sum_{\ell < \ell_0} B^{\ell_0 - \ell} \cdot \frac{B^{2\ell}}{4} \leq \frac{4}{B-1}.
\end{aligned}$$

Consequently, it holds that

$$\begin{aligned}
&\mathbb{E}_{M_{j-1}, M_j} [\text{TV}(P_{Z_j}, P_{Z_j \mid M_{j-1}, M_j, E_{j,1}})] \\
&= \mathbb{E}_{M_{j-1}, M_j} [\text{TV}(P_{Z_j \mid M_{j-1}}, P_{Z_j \mid M_{j-1}, M_j, E_{j,1}})] \\
&\leq \mathbb{E}_{M_{j-1}, M_j} [\text{TV}(P_{Z_j \mid M_{j-1}}, P_{Z_j \mid M_{j-1}, M_j}) + \text{TV}(P_{Z_j \mid M_{j-1}, M_j}, P_{Z_j \mid M_{j-1}, M_j, E_{j,1}})] \\
&= \mathbb{E}_{M_{j-1}, M_j} [\text{TV}(P_{Z_j \mid M_{j-1}}, P_{Z_j \mid M_{j-1}, M_j})] + \mathbb{E}_{M_{j-1}, M_j} [\mathbb{P}(E_{j,1}^c \mid M_{j-1}, M_j)] \\
&\leq \sqrt{\frac{1}{2} \mathbb{E}_{M_{j-1}, M_j} [\text{KL}(P_{Z_j \mid M_{j-1}} \parallel P_{Z_j \mid M_{j-1}, M_j})]} + \frac{4}{B-1} \\
&= \sqrt{I(Z_j; M_j \mid M_{j-1})} + \frac{4}{B-1} \\
&\leq \sqrt{\alpha} + \frac{4}{B-1},
\end{aligned}$$

### 33:10 On the Amortized Complexity of Approximate Counting

which can be made small by choosing  $\alpha > 0$  small enough and  $B \in \mathbb{N}$  large enough. Note that in the above inequality we have used the triangle inequality  $\text{TV}(P, Q) \leq \text{TV}(P, R) + \text{TV}(Q, R)$ , the conditioning relationship  $\text{TV}(P, P|_E) = P(E^c)$ , Pinsker's inequality  $\text{TV}(P, Q) \leq \sqrt{\text{KL}(P||Q)}/2$ , and Jensen's inequality  $\mathbb{E}[\sqrt{X}] \leq \sqrt{\mathbb{E}[X]}$ .

The next good event  $E_2$  concerns the simultaneous occurrence of  $\{E_{j,1}\}$  for a constant proportion of  $j \in J_0$ . Specifically,  $E_2$  is the event that there exists some  $J_1 \subseteq J_0$  such that:

1.  $|J_1| \geq |J_0|/2 \geq T/(4B^{\ell_0})$ ;
2. event  $E_{j,1}$  is true for all  $j \in J_1$ ;
3. a small TV distance  $\text{TV}(P_{Z_j}, P_{Z_j|M_{j-1}, M_j, E_{j,1}}) \leq 1/4$  (denoted by event  $E_{j,2}$ ) for all  $j \in J_1$ .

Since  $\{(Z_j^<, Z_j, Z_j^>)\}$  are conditionally independent given  $\{M_j\}$ ,

$$\begin{aligned} & \mathbb{E}_{\{M_j\}} \left[ \sum_{j \in J_0} \mathbb{1}(E_{j,1} \cap E_{j,2}) \right] \\ &= \sum_{j \in J_0} \mathbb{E}_{M_{j-1}, M_j} [\mathbb{1}(E_{j,1} \cap E_{j,2})] \\ &\geq \sum_{j \in J_0} \left( 1 - \mathbb{E}_{M_{j-1}, M_j} [\mathbb{P}(E_{j,1}^c | M_{j-1}, M_j)] \right. \\ &\quad \left. - 4 \cdot \mathbb{E}_{M_{j-1}, M_j} [\text{TV}(P_{Z_j}, P_{Z_j|M_{j-1}, M_j, E_{j,1}})] \right) \\ &\geq \left( 1 - \frac{4}{B-1} - 4 \left( \sqrt{\alpha} + \frac{4}{B-1} \right) \right) |J_0| \\ &\geq \frac{3}{4} |J_0|, \end{aligned}$$

by choosing  $\alpha$  small enough and  $B$  large enough. Consequently, by Markov's inequality, we have  $\mathbb{P}(E_2) \geq 1/2$  over the randomness of  $\{M_j\}$  and  $\{(Z_j^<, Z_j, Z_j^>)\}$ .

Now we condition on  $E_2$  and arrive at the desired contradiction. For a probability distribution  $P$  over the real line and  $\Delta \geq 0$ , define a quantity  $f(P, \Delta)$  as follows:

$$f(P, \Delta) = \max\{\delta > 0 : \exists L \in \mathbb{R} \text{ such that } P((-\infty, L]) \geq \delta, P([L + \Delta, \infty)) \geq \delta\}.$$

Intuitively, a *small*  $f(P, \Delta)$  implies that the distribution  $P$  assigns a lot of probability to *some* interval of length  $\Delta$ . The following lemma summarizes some properties of  $f(P, \Delta)$ .

► **Lemma 7.** *Let  $P$  and  $Q$  be two probability distributions over  $\mathbb{R}$ , and  $P \star Q$  denote their convolution. For  $\Delta_1, \Delta_2, \Delta \geq 0$ , it holds that*

$$\begin{aligned} f(P \star Q, \Delta_1 + \Delta_2) &\geq f(P, \Delta_1) f(Q, \Delta_2), \\ f(P \star Q, \Delta) &\geq f(P, \Delta)/2. \end{aligned}$$

**Proof.** For the first claim, suppose that

$$\begin{aligned} \min\{P((-\infty, L_1]), P([L_1 + \Delta_1, \infty))\} &\geq f(P, \Delta_1), \\ \min\{Q((-\infty, L_2]), Q([L_2 + \Delta_2, \infty))\} &\geq f(Q, \Delta_2). \end{aligned}$$

Then the first inequality follows from

$$\begin{aligned} P \star Q((-\infty, L_1 + L_2]) &\geq P((-\infty, L_1]) Q((-\infty, L_2]) \geq f(P, \Delta_1) f(Q, \Delta_2), \\ P \star Q([L_1 + L_2 + \Delta_1 + \Delta_2, \infty)) & \\ &\geq P([L_1 + \Delta_1, \infty)) Q([L_2 + \Delta_2, \infty)) \geq f(P, \Delta_1) f(Q, \Delta_2). \end{aligned}$$

The second claim is a direct consequence of the first claim and  $f(Q, 0) \geq 1/2$ . ◀



To apply Lemma 7, we consider the conditional distribution  $P_{S|\{M_j\}}$ , where  $S = \sum_{i=1}^n X_i$  is the total number of counter updates, and  $\{M_j\}$  are the memory states at scale  $\ell_0$  defined before. Since the counter  $\mathcal{A}$  is  $(\epsilon, \delta)$ -approximate, in expectation  $P_{S|\{M_j\}}$  must have probability mass at least  $1 - \delta$  on an interval of size  $2\epsilon n$ . This implies that

$$\mathbb{E}_{\{M_j\}} [f(P_{S|\{M_j\}}, 2\epsilon n)] \leq \delta. \quad (1)$$

On the other hand, since  $\{(Z_j^<, Z_j, Z_j^>)\}$  are conditionally independent given  $\{M_j\}$ , we may invoke Lemma 7 (first part for  $J_1$  and second part for  $J_1^c$ ) to arrive at

$$\begin{aligned} \mathbb{E}_{\{M_j\}} [f(P_{S|\{M_j\}}, 2\epsilon n)] &\geq \mathbb{P}(E_2) \mathbb{E}_{\{M_j\}} [f(P_{S|\{M_j\}}, 2\epsilon n) \mid E_2] \\ &\geq \frac{1}{2} \cdot \mathbb{E}_{\{M_j\}} \left[ \prod_{j \in J_1} f(P_{\text{sum}(Z_j^<, Z_j, Z_j^>)|M_{j-1}, M_j, \frac{2\epsilon n}{|J_1|}}) \cdot \left(\frac{1}{2}\right)^{\frac{T}{B^{\ell_0}} - |J_1|} \mid E_2 \right] \\ &\geq \frac{1}{2T} \cdot \mathbb{E}_{\{M_j\}} \left[ \prod_{j \in J_1} f(P_{\text{sum}(Z_j^<, Z_j, Z_j^>)|M_{j-1}, M_j, 8\epsilon L B^{\ell_0}}) \mid E_2 \right]. \end{aligned}$$

Conditioned on the event  $E_2$ , the event  $E_{j,1}$  implies that the deviation of  $\text{sum}(Z_j^<)$  to its posterior mean is at most  $B^{\ell_0}/4$ , and the event  $E_{j,2}$  implies that the posterior (marginal) distribution of  $Z_j$  puts at least  $1/4$  probability mass on both 0 and  $B^{\ell_0}$ . Moreover,  $\text{sum}(Z_j^>)$  is always an integral multiple of  $B^{\ell_0+1}$ . Now we prove that

$$f(P_{\text{sum}(Z_j^<, Z_j, Z_j^>)|M_{j-1}, M_j, E_{j,1}, E_{j,2}}, \frac{B^{\ell_0}}{3}) \geq \frac{1}{16}.$$

We distinguish into two cases:

1. Case I:  $\mathbb{P}(\text{sum}(Z_j^>) \neq \text{median}(\text{sum}(Z_j^>))) \geq 1/8$ . As  $\text{sum}(Z_j^>)$  is always an integral multiple of  $B^{\ell_0+1}$ , this implies that with probability at least  $\frac{1}{8}$ ,  $|\text{median}(\text{sum}(Z_j^>)) - \text{sum}(Z_j^>)|$  is at least  $B^{\ell_0+1}/2$ . By symmetry, without loss of generality we may assume that  $\text{median}(\text{sum}(Z_j^>)) - \text{sum}(Z_j^>)$  is at least  $B^{\ell_0+1}/2$  with probability at least  $(1/8)/2 = 1/16$ . Moreover, the range of  $Z_j$  is  $B^{\ell_0}$ , and the range of  $\text{sum}(Z_j^<)$  is at most  $B^{\ell_0}/2$  under  $E_{j,1}$ . Consider the interval

$$\left[ \text{median}(\text{sum}(Z_j^<, Z_j, Z_j^>)), \text{median}(\text{sum}(Z_j^<, Z_j, Z_j^>)) + \frac{B^{\ell_0}}{3} \right]$$

of length  $B^{\ell_0}/3$ , it is clear that

$$\begin{aligned} \mathbb{P}(\text{sum}(Z_j^<, Z_j, Z_j^>) \leq \text{median}(\text{sum}(Z_j^<, Z_j, Z_j^>))) &\geq \frac{1}{2}, \\ \mathbb{P}\left(\text{sum}(Z_j^<, Z_j, Z_j^>) \geq \text{median}(\text{sum}(Z_j^<, Z_j, Z_j^>)) + \frac{B^{\ell_0}}{3}\right) \\ &\geq \mathbb{P}\left(\text{sum}(Z_j^>) \geq \text{median}(\text{sum}(Z_j^>)) + \frac{3B^{\ell_0}}{2} + \frac{B^{\ell_0}}{3}\right) \geq \frac{1}{16}, \end{aligned}$$

as long as  $B^{\ell_0+1}/2 \geq 11B^{\ell_0}/6$ , or equivalently  $B \geq 4$ .

2. Case II:  $\mathbb{P}(\text{sum}(Z_j^>) = \text{median}(\text{sum}(Z_j^>))) \geq 7/8$ . In this case, we argue that each of the following two probabilities is at least  $1/16$ :

$$\mathbb{P}\left(\text{sum}(Z_j^<) + Z_j + \text{sum}(Z_j^>) \leq \mathbb{E}[\text{sum}(Z_j^<)] + \frac{B^{\ell_0}}{3} + \text{median}(\text{sum}(Z_j^>))\right),$$

### 33:12 On the Amortized Complexity of Approximate Counting

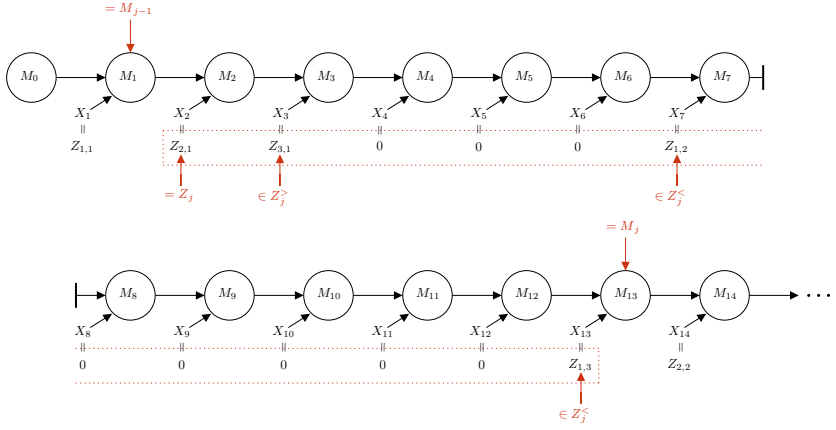
$$\mathbb{P} \left( \text{sum}(Z_j^<) + Z_j + \text{sum}(Z_j^>) \geq \mathbb{E}[\text{sum}(Z_j^<)] + \frac{2B^{\ell_0}}{3} + \text{median}(\text{sum}(Z_j^>)) \right).$$

By symmetry we only prove the first claim, where the event occurs whenever  $Z_j = 0$  and  $\text{sum}(Z_j^>)$  is equal to its median. By the union bound, this happens with probability at least  $1 - (1/8 + 3/4) = 1/8$ .

Consequently, as long as  $\epsilon \leq 1/(24L) = O(1/\log \log(1/\delta))$ ,

$$\mathbb{E}_{\{M_j\}} [f(P_{S|\{M_j\}}, 2\epsilon n)] \geq \left( \frac{1}{32} \right)^T. \quad (2)$$

However, as  $T < \log_{32}(1/\delta)$ , inequalities (1) and (2) are contradictory to each other. Thus, the assumption that  $\text{IC}(\mathcal{A}; \mathcal{D}) < \epsilon n L$  cannot be true.  $\blacktriangleleft$



■ **Figure 3** A simple example that illustrates the interleaving process for our random stream. We set the parameters  $B = 2$  and  $T = 8$  so that we sample from  $L = 3$  scales to get a stream of total length  $n = 24$ . We do not illustrate the entire stream to save space. In this example we consider  $\ell_0 = 2$  and demonstrate what a typical  $Z_j$  looks like. The sequence of random variables inside the red dotted box/window consists of the terms  $(Z_j^<, Z_j, Z_j^>)$  and we highlight the memory states  $M_{j-1}, M_j$ .

## 2.2 Amortized space complexity via direct sum

We will now “lift” the information lower bound for any  $(\epsilon, \delta)$ -approximate counter to a lower bound for any  $(k, \epsilon, \delta)$ -approximate counter from which we can derive a lower bound on the memory size. The proof follows from a simple direct sum argument applied to the information cost.

► **Lemma 8.** *Any  $(k, \epsilon, \delta)$ -approximate counter with parameters  $\delta \in (0, c_1)$  and  $\epsilon \in (0, \frac{c_2}{\log \log(1/\delta)})$  that maintains  $k$  counters with total sum at most  $n \geq k(c_3 \log^{c_4}(1/\delta))$  must use*

$$\Omega(k \log \log(1/\delta)) = \Omega(k \min\{\log(n/k), \log \log(1/\delta)\})$$

*bits of memory in expectation.*

**Proof.** We prove the lower bound via a reduction to the information lower bound for a single counter. That is, we will embed updates from the hard distribution  $\mathcal{D}$  used to prove the single counter lower bound in Lemma 6 into the  $k$  counter problem. Fix a  $(k, \epsilon, \delta)$ -approximate counter  $\mathcal{A}_k$ . Since the distribution  $\mathcal{D}$  is a product distribution, we can write  $\mathcal{D} = \prod_{i=1}^{n'} \mathcal{D}_i$  for  $n' := n/k$ . We define  $X_i^k := (X_{i,1}, \dots, X_{i,k}) \sim (\mathcal{D}_i)^k$  for each  $i \in [n']$ . We consider  $X_i^k$  to be the  $i$ th input to  $\mathcal{A}_k$  where  $X_{i,j}$  is the update applied to  $j$ th counter. The overall input to  $\mathcal{A}_k$  is

$$(X_1^k, \dots, X_{n'}^k) \sim \prod_{i=1}^{n'} (\mathcal{D}_i)^k,$$

and the total sum of all counts is at most  $n = kn' \geq k(c_3 \log^{c_4}(1/\delta))$  (see Lemma 6). Define  $\mathcal{D}_{\text{int}} := \prod_{i=1}^{n'} (\mathcal{D}_i)^k$ . Let  $M_i$  be the memory state of the algorithm after processing the  $i$ th input  $X_i^k$ . We can lower bound the information cost incurred by  $\mathcal{A}_k$ :

$$\text{IC}(\mathcal{A}_k, \mathcal{D}_{\text{int}}) = \sum_{i=1}^{n'} \sum_{j=1}^i I(M_i; X_j^k \mid M_{j-1}) \geq \sum_{u=1}^k \sum_{i=1}^{n'} \sum_{j=1}^i I(M_i; X_{j,u} \mid M_{j-1}),$$

where the inequality follows from the superadditivity of conditional mutual information (Proposition 4).

We now explain how we embed the single counter problem into the  $k$  counter problem using  $\mathcal{A}_k$ . Given the input  $X^{n'} = (X_1, \dots, X_{n'}) \sim \mathcal{D}$  for a single counter, we pick an index  $u \in [k]$  uniformly at random and proceed as if the updates are applied to  $u$ th counter of  $\mathcal{A}_k$ . Denote by  $U$  this uniformly random index. For the other counters, we simulate “fake” updates from the same distribution and apply them to  $\mathcal{A}_k$  as if they were received as inputs. Denote by  $\mathcal{A}'$  the resulting approximate counter that maintains  $X^{n'}$ . We can upper bound the information cost of  $\mathcal{A}'$  by

$$\begin{aligned} \text{IC}(\mathcal{A}', \mathcal{D}) &= \sum_{i=1}^{n'} \sum_{j=1}^i I(M_i; X_{j,U} \mid M_{j-1}, U) \\ &= \frac{1}{k} \sum_{u=1}^k \sum_{i=1}^{n'} \sum_{j=1}^i I(M_i; X_{j,u} \mid M_{j-1}) \\ &\leq \frac{1}{k} \sum_{i=1}^{n'} \sum_{j=1}^i I(M_i; X_j^k \mid M_{j-1}) \\ &= \frac{\text{IC}(\mathcal{A}_k, \mathcal{D}_{\text{int}})}{k}. \end{aligned}$$

Combining the above with Lemma 6 provides a lower bound on  $\text{IC}(\mathcal{A}', \mathcal{D})$ , which further implies the space lower bound

$$\frac{1}{n'} \sum_{i=1}^{n'} \mathbb{E}[|M_i|] \geq \Omega(k \min\{\log(n/k), \log \log(1/\delta)\})$$

for the claimed range of  $\epsilon$  and  $\delta$ . ◀

### 2.3 Offline lower bound

We now state and prove the remaining part of the lower bound.

### 33:14 On the Amortized Complexity of Approximate Counting

► **Lemma 9.** For any  $\epsilon \in (0, c_1)$ ,  $\delta \in (0, c_2)$  and  $1 < k \leq n$ , any  $(k, \epsilon, \delta)$ -approximate counter  $\mathcal{A}$  that maintains  $k$  counters with total sum at most  $n$  must use

$$|M| = \Omega(\min\{k \log(n/k), k \log(1/\epsilon) + k \log \log(n/k)\})$$

bits of memory.

**Proof.** Let  $N(i) = \lceil (e^{4i\epsilon} - 1) \cdot \epsilon^{-1} \rceil$ . Some simple calculations show

$$\begin{aligned} & (1 - \epsilon) \cdot N(i + 1) - (1 + \epsilon) \cdot N(i) \\ & \geq \epsilon^{-1}(1 - \epsilon)(e^{4(i+1)\epsilon} - 1) - \epsilon^{-1}(1 + \epsilon)(e^{4i\epsilon} - 1) - (1 + \epsilon) \\ & = \epsilon^{-1}((1 - \epsilon)e^\epsilon - 1 - \epsilon)e^{4i\epsilon} + 1 - \epsilon \\ & \geq \epsilon^{-1}((1 - \epsilon)e^{4\epsilon} - 1 - \epsilon)e^{4i\epsilon} \\ & \geq \epsilon^{-1}((1 - \epsilon)(1 + 4\epsilon) - 1 - \epsilon) \cdot 1 \\ & = 2 - 4\epsilon > 0. \end{aligned}$$

So for any  $i \neq i'$ ,  $N(i)$  and  $N(i')$  must have a  $(1 \pm \epsilon)$  multiplicative gap. We will consider  $k$  counters that take on such values, i.e.  $k$  counters that receive an (arbitrary) sequence of increments leading to counts  $N(i_1), N(i_2), \dots, N(i_k)$  respectively. It is easy to see that if  $i_r \leq (4\epsilon)^{-1} \cdot \ln(1 + n\epsilon/k)$  for all  $r \in [k]$ , the total sum of all the counters will be at most  $n$ . Let  $q := \lfloor (4\epsilon)^{-1} \cdot \ln(1 + n\epsilon/k) \rfloor$  and define the set of possible counts  $\mathbf{N} := \{N(1), N(2), \dots, N(q)\}$ . We can represent the counts of all  $k$  counters as vectors in  $\mathbf{N}^k$ .

Consider a “large” collection of vectors  $V \subseteq \mathbf{N}^k$  such that for every pair of counts  $u, v \in V$ ,  $u$  and  $v$  differ in at least at least 90% of the coordinates. Notice that this implies that  $u$  and  $v$  differ multiplicatively in at least 90% of the coordinates by definition. Such a  $V$  is equivalent to an error correcting code.

► **Definition 10.** An error correcting code  $C$  of length  $k$  over a finite alphabet  $\Sigma$  is a subset of  $\Sigma^k$ . The elements of  $C$  are called code words. The distance of the code  $C$ , denoted  $\Delta(C)$ , is defined as the minimum hamming distance between any two code words  $c_1, c_2 \in C$ , i.e.

$$\Delta(C) := \min_{\substack{c, c' \in C \\ c \neq c'}} \Delta(c, c'),$$

where  $\Delta(c, c') := |\{i : c_i \neq c'_i\}|$  is the hamming distance between two vectors.

In the language of error correcting codes, we want our collection of counts  $V$  to be a large error correcting code with a minimum distance of  $0.9k$ . Fortunately, the Gilbert-Varshamov bound immediately implies the existence of such a  $V$  that is large enough for our purposes.

► **Lemma 11 (Gilbert-Varshamov bound).** For any alphabet size  $q > 1$ , code length  $k$  and distance  $d \leq k$ , there exists an error correcting code  $C$  with size,

$$|C| \geq \frac{q^k}{\sum_{i=0}^{d-1} \binom{k}{i} (q-1)^i}.$$

Consequently, for  $d = 0.9k$  and any  $q$  larger than a universal constant, there is a code  $C$  with size  $|C| \geq q^{0.05k}$ .

Thus, when  $q$  is a large enough constant (which can be achieved by making for  $\epsilon$  smaller than some universal constant  $c$ ), the Gilbert-Varshamov bound tells us there is a choice of  $V$  such that  $|V| \geq q^{0.05k}$ . Fix a  $(k, \epsilon, \delta)$ -approximate counter  $\mathcal{A}$ . For any  $v \in V$  and  $i \in [k]$ ,  $\mathcal{A}$  must

accurately approximate  $v_i$  with probability at least  $1 - \delta$ . Denote the event that  $\mathcal{A}$  correctly approximates  $v_i$  by  $E_{v,i}$ . Since  $\mathbb{E}[\sum_{i=1}^k \mathbb{1}(E_{v,i}^c)] \leq k \cdot \delta < k/20$ , by Markov's inequality we can conclude the existence of a subset  $I_v \subseteq [k]$  such that:

1.  $|I_v| \geq 0.9k$ ,
2.  $\mathbb{P}[\cap_{i \in I_v} E_{v,i}] > 1/2$ . Put in words, with probability at least  $1/2$ , for every  $i \in I_v$ , the algorithm  $\mathcal{A}$  outputs a  $(1 + \epsilon)$ -approximation for  $v_i$ .

Define the event  $E_v := \cap_{i \in I_v} E_{v,i}$ . Since  $\mathbb{E}[\sum_{v \in V} \mathbb{1}(E_v^c)] \leq |V|/2$ , a standard averaging argument implies the existence of a fixed choice for the random seed of  $\mathcal{A}$  and a subset of counts  $V' \subseteq V$  such that:

1.  $|V'| > |V|/2$ ,
2. and the algorithm is correct on all  $v \in V'$  in the sense of the event  $E_v$  on this random seed.

Fix such a random seed and denote the now deterministic algorithm  $\mathcal{A}'$ . For two counts  $u, v \in V'$  define the set  $D_{u,v} := \{i \in [k] : u_i \neq v_i\}$ . By construction we have that  $|D_{u,v}| \geq 0.9k$ . We have

$$\begin{aligned} |D_{u,v} \cap I_u \cap I_v| &\geq k - |D_{u,v}^c| - |I_u^c| - |I_v^c| \\ &\geq k - 0.3k = 0.7k > 1. \end{aligned}$$

Thus, there is at least one index  $i^* \in [k]$  such that  $u_{i^*} \neq v_{i^*}$  and  $\mathcal{A}'$   $(1 + \epsilon)$ -approximates both  $u_{i^*}$  and  $v_{i^*}$ , so  $\mathcal{A}'$  arrives at a different state for  $u$  and  $v$ . Since this holds for every pair in  $V'$ , we can conclude that  $\mathcal{A}'$  must arrive at a different state for every count in  $V'$ . We can now conclude that

$$2^{|M|} \geq |V'| \geq 0.5 \cdot (\lfloor (4\epsilon)^{-1} \cdot \ln(1 + n\epsilon/k) \rfloor)^{0.05k} = \left( \Omega\left(\frac{\ln(1 + n\epsilon/k)}{\epsilon}\right) \right)^{0.05k},$$

or

$$|M| \geq 0.05k \log\left(\frac{\ln(1 + n\epsilon/k)}{\epsilon}\right) - O(k).$$

We distinguish into three cases:

1. If  $\epsilon < k/n$ , then  $\frac{\ln(1+n\epsilon/k)}{\epsilon} = \Omega(n/k)$ , which implies  $|M| \geq 0.05k \log(n/k) - O(k)$ .
2. If  $k/n \leq \epsilon < \sqrt{k/n}$ , we have

$$\begin{aligned} |M| &\geq 0.05k \log(1/\epsilon) - O(k) \\ &\geq 0.05k \log(1/\epsilon) + k \log \log(n/k) - O(k \log \log(1/\epsilon)). \end{aligned}$$

3. If  $\epsilon > \sqrt{k/n}$ , we have

$$\begin{aligned} |M| &\geq 0.05k \log(1/\epsilon) + 0.05k \log \log(en/k) - O(k) \\ &\geq 0.05k \log(1/\epsilon) + 0.05k \log \log(n/k) - O(k). \end{aligned}$$

Putting all the above bounds together gives us

$$\begin{aligned} |M| &\geq \min\{0.05k \log(n/k) - O(k), 0.05k \log(1/\epsilon) + 0.025k \log \log(n/k) \\ &\quad - O(k \log \log(1/\epsilon))\} \\ &= \Omega(\min\{k \log(n/k), k \log(1/\epsilon) + k \log \log(n/k)\}). \end{aligned}$$

◀

### 3 Upper bounds

We now state matching upper bounds  $(k, \epsilon, \delta)$ -approximate counting that follow immediately from the single counter case. We give upper bounds in two regimes: the case  $k \leq N$ , and the case  $k > N$ . We start by analyzing the first case.

Recall that the space usage of an approximate counter is typically a random variable, and the goal is to then prove an upper bound on the *expected* space (or, say, a high probability bound). The work of Nelson and Yu [9] provided the following bound on expected space usage for a single counter:

► **Theorem 12** ([9]). *For any  $\epsilon, \delta \in (0, 1/2)$ , there is an  $(\epsilon, \delta)$ -approximate counter with expected space usage  $O(\log \log N + \log \log(1/\delta) + \log(1/\epsilon))$  bits.*

The following is then a very simple corollary of Theorem 12.

► **Corollary 13.** *For any  $\epsilon, \delta \in (0, 1/2)$  and  $1 \leq k \leq N$ , there is a  $(k, \epsilon, \delta)$ -approximate counter that uses  $O(k(\log \log(2N/k) + \log \log(1/\delta) + \log(1/\epsilon)))$  bits in expectation.*

**Proof.** We simply instantiate  $k$  independent copies of the data structure from Theorem 12 to provide  $k$  independent approximate counters, one per actual counter. For each  $1 \leq i \leq k$ , let  $S_i$  denote the (random) number of bits of memory used to store the approximate counter representing the  $i$ th counter  $N_i$  and recall  $N := \sum_i N_i$ . Then the total expected space is

$$\begin{aligned} \mathbb{E} \left[ \sum_{i=1}^k S_i \right] &= \sum_{i=1}^k \mathbb{E}[S_i] \\ &\leq C \sum_{i=1}^k (\log \log(N_i) + \log \log(1/\delta) + \log(1/\epsilon)) \quad (\text{Theorem 12}) \\ &= Ck(\log \log(1/\delta) + \log(1/\epsilon)) + \sum_{i=1}^k \log \log(N_i) \\ &\leq Ck(\log \log(4N/k) + \log \log(1/\delta) + \log(1/\epsilon)) \quad (\text{Jensen}), \end{aligned}$$

where the last inequality uses concavity of the function  $x \in (1, \infty) \mapsto \log \log(x)$ . Note that we inject a constant 4 in the iterated logarithm so that the  $\log \log$  term stays nonnegative. ◀

We now turn to the case  $k > N$ . In this case, there is a clear lower bound of  $\Omega(N \log(Ck/N))$  bits, since  $\log(\sum_{j=1}^N \binom{k}{j})$  bits of memory are needed to simply remember which counters are non-zero, and the logarithm of this sum is  $\Theta(N \log(Ck/N))$  [5, Exercise 9.42]. We claim that this is also an upper bound. Specifically, we can use  $O(N \log(Ck/N))$  bits to remember the locations of the  $t \leq N$  non-zero counters. Now we have reduced to the case  $k = t \leq N$  and can apply Corollary 13 to approximately remember their values.

---

#### References

- 1 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.
- 2 Mark Braverman, Sumegha Garg, and David P. Woodruff. The coin problem with applications to data streams. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 318–329, 2020.

- 3 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 270–278, 2001.
- 4 Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 2 edition, 2006. URL: <http://www.elementsofinformationtheory.com/>.
- 5 Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, 1994.
- 6 Elena Kolevska. What happens with Redis runs out of memory, December 2018. URL: <https://www.youtube.com/watch?v=Xjq5XL2u3po>.
- 7 Jérémie Lumbroso. The story of HyperLogLog: How Flajolet processed streams with coin flips. *CoRR*, abs/1805.00612, 2018.
- 8 Robert H. Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978.
- 9 Jelani Nelson and Huacheng Yu. Optimal bounds for approximate counting. In *Proceedings of the 41st ACM International Conference on Principles of Database Systems (PODS)*, pages 119–127, 2022.
- 10 Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.







# Matrix Multiplication Reductions

Ashish Gola ✉

Simon Fraser University, Burnaby, Canada

Igor Shinkar ✉ 

Simon Fraser University, Burnaby, Canada

Harsimran Singh ✉ 

Simon Fraser University, Burnaby, Canada

---

## Abstract

In this paper we study a worst case to average case reduction for the problem of matrix multiplication over finite fields. Suppose we have an efficient *average case* algorithm, that given two random matrices  $A, B$  outputs a matrix that has a non-trivial correlation with their product  $A \cdot B$ . Can we transform it into a *worst case* algorithm, that outputs the correct answer for all inputs without incurring a significant overhead in the running time? We present two results in this direction.

**Two-sided error in the high agreement regime.** We begin with a brief remark about a reduction for high agreement algorithms, i.e., an algorithm which agrees with the correct output on a large (say  $> 0.9$ ) fraction of entries, and show that the standard self-correction of linearity allows us to transform such algorithms into algorithms that work in worst case.

**One-sided error in the low agreement regime.** Focusing on average case algorithms with one-sided error, we show that over  $\mathbb{F}_2$  there is a reduction that gets an  $O(T)$  time *average case* algorithm that given a random input  $A, B$  outputs a matrix that agrees with  $A \cdot B$  on at least 51% of the entries (i.e., has only a slight advantage over the trivial algorithm), and transforms it into an  $\tilde{O}(T)$  time *worst case* algorithm, that outputs the correct answer for *all inputs* with high probability.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** Matrix Multiplication, Reductions, Worst case to average case reductions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.34

**Category** RANDOM

**Related Version** *arXiv*: <https://arxiv.org/pdf/2404.08085>

**Acknowledgements** We are grateful to the anonymous referees for their helpful comments. We also thank Sasha Golovnev and Tom Gur for their valuable feedback.

## 1 Introduction

The problem of efficiently multiplying two matrices has been extensively studied for decades. Improving on the straightforward  $O(n^3)$  time algorithm, Strassen's algorithm [24] computes the product of two matrices in time  $O(n^{\log_2 7} = n^{2.807})$ , and it is perhaps the most widely used in practice. Since then, a long and exciting line of research ([19, 5, 21, 20, 23, 9, 22, 25, 17, 1]) has led to a significant improvement of the value of the optimal exponent of the running time for matrix multiplication problem. The fastest algorithm known today is due to Duan, Wu, and Zhou [10], and its running time is  $O(n^{2.371866})$ .

Worst-case to average-case reductions serve as a means to convert algorithms that output correct answers on a fraction of inputs into algorithms with correct outputs on all possible inputs. These reductions can be viewed from two different perspectives. From the hardness point of view, they can be used to show that a problem maintains its hardness even in the



© Ashish Gola, Igor Shinkar, and Harsimran Singh;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 34; pp. 34:1–34:15



Leibniz International Proceedings in Informatics

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

## 34:2 Matrix Multiplication Reductions

average case. From the algorithmic side, they provide a framework for developing worst-case algorithms, by first designing weak algorithms with average case guarantees, and then transforming them into algorithms which work on all outputs.

In this paper, we study the following variant of a worst-case to average-case reduction for the matrix multiplication problem. Suppose we have an efficient algorithm that given two random matrices  $A, B \in \mathbb{F}^{n \times n}$ , computes a matrix  $C \in \mathbb{F}^{n \times n}$  that agrees with the product  $A \cdot B$  on a large fraction of the entries of the matrix. Can we transform such an algorithm into one that computes  $A \cdot B$  correctly for all entries of the output matrix without incurring a significant overhead in the running time?

More formally, we define the *agreement* between two matrices as the fraction of entries on which the two matrices agree.

► **Definition 1.** Let  $\mathbb{F}$  be a field, and let  $A, B \in \mathbb{F}^{n \times n}$  be two matrices. We define *agreement* between  $A$  and  $B$ , denoted by  $\text{agr}(A, B)$ , as the fraction of entries  $(i, j)$  on which  $A_{i,j} = B_{i,j}$ , i.e.,

$$\text{agr}(A, B) = \frac{|\{(i, j) : A_{i,j} = B_{i,j}\}|}{n^2} .$$

Then, our goal can be stated as the task of transforming an algorithm that on a random input  $A, B$  outputs a matrix  $C$  such that  $\text{agr}(C, AB) \geq \alpha$  for some parameter  $\alpha \in [0, 1]$  into an algorithm that solves the matrix multiplication problem *correctly on all inputs*.

We present two results in this direction. Both results consider the matrix multiplication problem over finite fields.

### High agreement regime with two-sided error

We show that any algorithm that solves the matrix multiplication problem correctly on a high fraction of the coordinates, can be converted into a worst case algorithm. Specifically, we prove the following theorem.

► **Theorem 2.** Fix a finite field  $\mathbb{F}$ . Let  $\alpha \in (0, 1/8)$ . Let  $\text{ALG}$  be an algorithm that gets as input two matrices  $A, B \in \mathbb{F}^{n \times n}$ , runs in time  $T(n)$ , and outputs a matrix  $\text{ALG}(A, B) \in \mathbb{F}^{n \times n}$ . Suppose that

$$\mathbb{E}_{A, B \in \mathbb{F}^{n \times n}} [\text{agr}(\text{ALG}(A, B), A \cdot B)] > 1 - \alpha .$$

Then, there is an algorithm  $\text{ALG}^*$  that gets as input two matrices  $A, B \in \mathbb{F}^{n \times n}$ , runs in time  $O(T(n) \cdot \log(n))$ , and outputs a matrix  $\text{ALG}^*(A, B) \in \mathbb{F}^{n \times n}$  such that for all  $A, B$  it holds that

$$\Pr[\text{ALG}^*(A, B) = A \cdot B] > 1 - 1/n ,$$

where the randomness is only over the internal coins of  $\text{ALG}^*$ .

The proof of this result relies on rather standard ideas, and essentially uses the self-correction of linear functions [6].

### Low agreement with one-sided error

For this result, we restrict our discussion to the finite field  $\mathbb{F}_2$ . Note that it is trivial to design an  $O(n^2)$  time algorithm such that  $\mathbb{E}_{A, B \in \mathbb{F}_2^{n \times n}} [\text{agr}(\text{ALG}(A, B), A \cdot B)] \geq 1/2$ . Indeed, the algorithm can simply output 0 in all entries irrespective of the input. Alternatively, the algorithm can output a random 0/1 matrix. Hence, it is natural to ask whether it is possible to obtain a better-than-1/2 algorithm for the matrix multiplication over  $\mathbb{F}_2$ .

Below we show that in the special case of *one-sided error* approximation, any better-than- $1/2$  approximation  $O(T)$  time algorithm can be transformed into a worst case algorithm with running time  $\tilde{O}(T)$ . Formally, we prove the following theorem.

► **Theorem 3.** *Let ALG be an algorithm that gets input two matrices  $A, B \in \mathbb{F}_2^{n \times n}$ , runs in time  $T(n)$ , and outputs a matrix  $\text{ALG}(A, B) \in \mathbb{F}_2^{n \times n}$ . Let  $\delta > 0$ , and suppose that*

- $\mathbb{E}_{A, B \in \mathbb{F}_2^{n \times n}}[\text{agr}(\text{ALG}(A, B), A \cdot B)] \geq 1/2 + \delta$ .
- If  $(AB)_{i,j} = 0$ , then  $\text{ALG}(A, B)_{i,j} = 0$ .

*Then, there is an algorithm  $\text{ALG}^*$  that gets as input two matrices  $A, B \in \mathbb{F}_2^{n \times n}$ , runs in time  $\tilde{O}(T(n))$ , and outputs a matrix  $\text{ALG}^*(A, B) \in \mathbb{F}_2^{n \times n}$  such that for all  $A, B$  it holds that*

$$\Pr[\text{ALG}^*(A, B) = A \cdot B] > 1 - 1/n,$$

where the randomness is only over the internal coins of  $\text{ALG}^*$ .

► **Remark 4.** Below we make several comments about Theorem 3.

1. Note that the conditions of the theorem can be written equivalently as follows.
  - $\Pr_{A, B \in \mathbb{F}_2^{n \times n}}[\text{ALG}(A, B)_{i,j} = 1] \geq \delta$ .
  - If  $(AB)_{i,j} = 0$ , then  $\text{ALG}(A, B)_{i,j} = 0$ .
2. The notion of algorithms with one-sided error is typically studied in the context of randomized algorithms, e.g., relating to the classes  $\mathcal{RP}$  (and  $\text{coRP}$ ), where the guarantee is that for *every NO input* the algorithm outputs the correct answer with probability 1, and for *every YES input* it is correct with probability at least  $2/3$ . The error model in Theorem 3 is different, as we consider algorithms that are correct on *random inputs* on all output 0-bits, and on at least some  $\alpha$ -fraction of 1-bits.
3. Alternatively, we can view the one-side error condition of Theorem 3 as an *errorless heuristic*, where in each entry of the matrix ALG outputs either 1 representing the correct answer, or says “I don’t know” and outputs 0.
4. We remark that the standard methods of self-correcting linear functions work in the high agreement regime, but fail when the average case guarantee is low. We apply the techniques from additive combinatorics developed in [2], particularly a version of the probabilistic Bogolyubov-Ruzsa Lemma, to perform a self-correction procedure which helps in this regime.
5. Our proof of Theorem 3 assumes that ALG is deterministic. It is rather straightforward to extend the proof and allow it to be randomized, by appropriately modifying the sets of good inputs  $(X_{i,j}$  and  $Y_{i,j}^A)$  to account for the randomness of the algorithm.

## 1.1 Related work

The study of average-case complexity began with Levin’s work [18], followed by subsequent works like [4]. A substantial body of research (e.g., [16], [15] and related references) identified numerous barriers in formulating worst-case to average-case reductions for NP-complete problems. For a comprehensive overview of this subject, see the classical surveys by Impagliazzo [14], Bogdanov and Trevisan [7] and Goldreich [12].

Asadi et al. [2, 3] presented a new framework for carrying out efficient worst-case to average case reductions for various fundamental problems. Particularly, for the problem of matrix multiplication, they proved that if there exists an  $O(T(n))$  time algorithm  $M$  for matrix multiplication which computes the correct output on an  $\epsilon$  fraction of inputs, then there exists a randomized algorithm  $M'$  which computes the correct output on all inputs, running in time  $O(\exp(O(\log^5(1/\epsilon))) \cdot T(n))$ . The proof relied on additive combinatorial techniques and used the probabilistic Bogolyubov-Ruzsa Lemma.

## 34:4 Matrix Multiplication Reductions

Hirahara and Shimizu [13] improved the  $\exp(O(\log^5(1/\epsilon)))$  overhead to an  $\tilde{O}(1/\epsilon)$  factor. Their idea involved dividing the output matrix into smaller blocks and using the Direct-Product Theorem in a black-box manner.

The aforementioned papers assume that we have access to an algorithm which gives a fully correct output on some fraction of the inputs, i.e., for these inputs *all entries* in the output matrix are correct. The setting presented in this paper, where the output of the given algorithm is not fully correct, seems to differ significantly from the works mentioned above. In particular, we do not see how to apply the Direct-Product theorem to our setting of the problem.

A related problem was studied by [11]. Specifically, they provided an  $\tilde{O}(n^2 + kn)$  time randomized algorithm and an  $\tilde{O}(kn^2)$  time deterministic algorithm for correcting the product of two matrices over a ring, where the product has at most  $k$  incorrect entries. Theorem 2 improves upon their work for a certain range of  $k$ , e.g.,  $n^2/20 < k < n^2/8$ , and Theorem 3 gives a new result for  $k$  closer to  $n^2/2$  in a very specific error model.

### 1.2 Open problems

We mention the following two problems that are left open in this work.

#### Low agreement with two-sided error

Is it possible to transform a two-sided error algorithm over  $\mathbb{F}_2$  with a low agreement guarantee into a worst case algorithm. That is, given an  $O(T(n))$  time algorithm ALG with the guarantee  $\mathbb{E}_{A,B \in \mathbb{F}_2^{n \times n}}[\text{agr}(\text{ALG}(A, B), A \cdot B)] > 1/2 + \delta$ , can we convert it into an algorithm that correctly outputs the correct answer on all inputs and has running time  $\tilde{O}(T(n))$ ?

#### Generalizing over finite fields

Extend Theorem 3 in a meaningful way to work over any finite field.

## 2 Preliminaries

For a positive integer  $n$  we define  $[n] = \{0, 1, \dots, n-1\}$ . We index the coordinates of our matrices starting from 0 rather than 1, which is typically more standard. We refer to the element in the row  $i$  and column  $j$  of the matrix  $A$  as  $A_{i,j}$ .

We define a notion of *row-shift* (or *row-rotation*) and *column-shift* as follows.

► **Definition 5.** Given a matrix  $A \in \mathbb{F}^{n \times m}$ ,  $0 \leq \pi \leq n-1$ , and  $0 \leq \sigma \leq m-1$ , define  $A^{\pi, \sigma}$  to be the matrix obtained from  $A$  by cyclically rotating all its rows downwards by  $\pi$  units and all its columns rightwards by  $\sigma$  units, that is,

$$(A^{\pi, \sigma})_{i,j} = A_{(i-\pi) \bmod n, (j-\sigma) \bmod m}$$

The following proposition is immediate from the definition above.

► **Proposition 6.** For any  $A, B, C \in \mathbb{F}^{n \times n}$  and any  $\pi, \sigma$  we have  $AB = C$  if and only if  $A^{\pi, 0} \cdot B^{0, \sigma} = C^{\pi, \sigma}$ .

## 2.1 Additive Combinatorics Tools

We now present the additive combinatorics toolkit which will be useful in the worst-case to average-case reduction for the low agreement regime with one-sided error.

For a set  $A \subseteq \mathbb{F}_2^n$ , let  $1_A: \mathbb{F}_2^n \rightarrow \{0, 1\}$  denote the indicator function of  $A$ . The Fourier expansion of a function  $f: \mathbb{F}_2^n \rightarrow \mathbb{C}$  is given by  $f(x) = \sum_{r \in \mathbb{F}_2^n} \hat{f}(r) \cdot \chi_r(x)$ , where  $\chi_r(x) = (-1)^{\langle x, r \rangle}$ , and the Fourier coefficients of  $f$  are defined as  $\hat{f}(r) = \langle f, \chi_r \rangle = \mathbb{E}_x[f(x) \cdot \chi_r(x)]$ . Parseval's identity says that  $\sum_{r \in \mathbb{F}_2^n} \widehat{1_A}(r)^2 = \langle 1_A, 1_A \rangle = \alpha$ , where  $\alpha$  is the density of  $A$ .

Define  $\text{Spec}_\gamma(A) = \{r \in \mathbb{F}_2^n : |\widehat{1_A}(r)| \geq \gamma\}$ . Below we state Chang's lemma, which describes a certain structure of  $\text{Spec}_\gamma(A)$ .

► **Lemma 7** (Chang's Theorem [8]). *Let  $A \subseteq \mathbb{F}^n$  be a set of size  $|A| = \alpha \cdot |\mathbb{F}|^n$ , and let  $\gamma > 0$ . Then*

$$\dim(\text{span}(\text{Spec}_{\gamma\alpha}(A))) \leq O\left(\frac{\log(1/\alpha)}{\gamma^2}\right).$$

Recall that the subset sum of two sets  $A$  and  $B$  is defined as  $A+B = \{a+b : a \in A, b \in B\}$ . Analogously, we define  $tA = A+A+\dots+A$  ( $t$  times) as  $tA = \{a_1+a_2+\dots+a_t : a_1, a_2, \dots, a_t \in A\}$ . The following lemma says that for an arbitrary set  $A \subseteq \mathbb{F}^n$ , the sumset  $tA$  contains a large affine subspace.

► **Lemma 8** (Probabilistic Bogolyubov-Ruzsa lemma). *Let  $A \subseteq \mathbb{F}_2^n$  be a set of size  $|A| = \alpha \cdot 2^n$ , for some  $\alpha \in (0, 1]$ , and let  $t \geq 3$  be an integer. Then,  $tA$  contains an affine subspace  $V \subseteq \mathbb{F}_2^n$  of dimension  $\dim(V) \geq n - O(\log(1/\alpha))$  such that for all  $v \in V$  it holds that*

$$\Pr_{a_1, a_2, \dots, a_{t-1} \in \mathbb{F}_2^n} [a_1, a_2, a_3, \dots, a_t \in A] \geq \alpha^t \left(1 + \frac{1}{2^{t-2}}\right) - \frac{\alpha^{t-1}}{2^{t-2}},$$

where  $a_t = v - a_1 - a_2 - \dots - a_{t-1}$ .

In particular, if  $t > \log_2(1/\alpha) + 2$ , then  $tA$  contains an affine subspace  $V \subseteq \mathbb{F}_2^n$  of dimension  $\dim(V) \geq n - k$ , for  $k = O(\log(1/\alpha))$ , such that for all  $v \in V$  it holds that

$$\Pr_{\substack{a_1, a_2, \dots, a_t \in \mathbb{F}_2^n \\ v = \sum_{i=1}^t a_i}} [a_1, a_2, a_3, \dots, a_t \in A] \geq (\alpha/2)^t.$$

Below we prove Lemma 8 only for odd values of  $t$ , which is slightly more complicated than the case of even  $t$ . After the proof, we remark how to modify the proof to work for even  $t$ 's.

**Proof.** Let  $A \subseteq \mathbb{F}_2^n$  be a set of size  $|A| = \alpha \cdot |\mathbb{F}|^n$ , for some  $\alpha \in (0, 1]$ . Consider the set

$$R = \text{Spec}_{\alpha/2} \setminus \{0\} = \{r \in \mathbb{F}_2^n \setminus \{0\} : |\widehat{1_A}(r)| > \frac{\alpha}{2}\}$$

Next we define an affine subspace  $V = \{v \in \mathbb{F}_2^n : \langle v, r \rangle = s_r \ \forall r \in R\}$  for some  $s_r \in \{0, 1\}$  to be defined later, and claim that  $V$  satisfies the conclusions of Lemma 8. We will need the following two claims.

▷ **Claim 9.** For all  $r \in R$  there exists  $s_r \in \{0, 1\}$  such that (1)  $\sum_{r \in R} \widehat{1_A}(r)^t \cdot (-1)^{s_r} \geq 0$  and (2) if  $r^* \in R$  is a linear combination  $r^* = \sum_{r \in R} c_r \cdot r$  of vectors in  $R$  (with  $c_r \in \mathbb{F}_2$ ), then  $s_{r^*} = \sum_{r \in R} c_r \cdot s_r \pmod{2}$ .

### 34:6 Matrix Multiplication Reductions

Proof. Let  $R'$  be a maximal subset of  $R$  of linearly independent vectors. Choose  $s_{r'} \in \{0, 1\}$  independently with probability 0.5 each for every  $r' \in R'$ . Now any  $r \in R \setminus R'$ , can be expressed as a linear combination  $r = \sum_{r'} c_{r'} \cdot r'$  of vectors in  $R'$  with  $c_{r'} \in \{0, 1\}$  define  $s_r = \sum_{r'} c_{r'} \cdot s_{r'}$ . It is immediate to verify that condition (2) is satisfied.

In order to satisfy condition (1) note that by linearity of expectation  $\mathbb{E}[\sum_{r \in R} \widehat{1}_A(r)^t \cdot (-1)^{s_r}] = 0$ , and hence there exists a choice of  $(s_r)_{r \in R}$  such that  $\sum_{r \in R} \widehat{1}_A(r)^t \cdot (-1)^{s_r} \geq 0$ , as required.  $\triangleleft$

$\triangleright$  Claim 10. We have  $\sum_{r \notin R, r \neq 0} |\widehat{1}_A(r)|^t \leq (\alpha/2)^{t-2}(\alpha - \alpha^2)$ .

Proof. For  $t \geq 3$ , it holds that

$$\begin{aligned} \sum_{r \notin R, r \neq 0} |\widehat{1}_A(r)|^t &\leq \max_{r \notin R, r \neq 0} |\widehat{1}_A(r)|^{t-2} \sum_{r \notin R, r \neq 0} |\widehat{1}_A(r)|^2 \\ &\leq (\alpha/2)^{t-2} \sum_{r \in \mathbb{F}_2^n \setminus \{0\}} \widehat{1}_A(r)^2 \\ &< (\alpha/2)^{t-2}(\alpha - \alpha^2) . \end{aligned} \quad \triangleleft$$

Define an affine subspace  $V = \{v \in \mathbb{F}_2^n : \langle v, r \rangle = s_r \ \forall r \in R\}$ , where  $s_r \in \{0, 1\}$  is from Claim 9. Note that if the vectors in  $R$  are linearly dependent, then the second condition of Claim 9 guarantees that we can define  $V = \{v \in \mathbb{F}_2^n : \langle v, r' \rangle = s_{r'} \ \forall r' \in R'\}$  for a maximal set  $R' \subset R$  of linearly independent vectors in  $R$ , and the remaining constraints will be satisfied by linearity. Then, according to Lemma 7 we have

$$\dim(V) \geq n - O(\log(1/\alpha)) .$$

Using the two claims above, and noting that  $\Pr_{a_1, a_2, \dots, a_{t-1} \in \mathbb{F}^n} [a_1, a_2, a_3, \dots, a_t \in A] = 1_A * 1_A * \dots * 1_A(v)$  ( $t$  times), for any  $v \in V$  we have

$$\begin{aligned} \Pr_{a_1, a_2, \dots, a_{t-1} \in \mathbb{F}^n} [a_1, a_2, a_3, \dots, a_t \in A] &= 1_A * 1_A * \dots * 1_A(v) \\ &= \sum_{r \in \mathbb{F}^n} \widehat{1}_A(r)^t \chi_r(v) \\ &= \widehat{1}_A(0)^t + \sum_{r \in R} \widehat{1}_A(r)^t \chi_r(v) + \sum_{r \notin R, r \neq 0} \widehat{1}_A(r)^t \chi_r(v) \\ &\geq \alpha^t + \sum_{r \in R} \widehat{1}_A(r)^t \cdot (-1)^{s_r} - (\alpha/2)^{t-2}(\alpha - \alpha^2) \\ &\geq \alpha^t + 0 - (\alpha/2)^{t-2}(\alpha - \alpha^2) \\ &= \alpha^t \left(1 + \frac{1}{2^{t-2}}\right) - \frac{\alpha^{t-1}}{2^{t-2}} . \end{aligned}$$

In particular, for  $t > \log_2(1/\alpha) + 2$ , we have

$$\begin{aligned} \Pr_{a_1, a_2, \dots, a_{t-1} \in \mathbb{F}^n} [a_1, a_2, a_3, \dots, a_t \in A] &\geq \alpha^t \left(1 + \frac{1}{2^{t-2}}\right) - \frac{\alpha^{t-1}}{2^{t-2}} \\ &\geq \alpha^t \left(1 + \frac{1}{2^{t-2}}\right) - \alpha^t \\ &\geq (\alpha/2)^t , \end{aligned}$$

as required.  $\blacktriangleleft$

► Remark 11. For even values of  $t$  the lemma is slightly easier. Specifically, since  $\widehat{1}_A(r)^t$  is always non-negative, we can take  $s_r = 0$  in Claim 9, and the rest of the proof works the same.

### 3 High Agreement with Two-Sided Error

In this section, we prove Theorem 2. Specifically, we show that if there exists an algorithm which, given two matrices  $A, B \in \mathbb{F}^{n \times n}$ , runs in time  $T(n)$  and correctly computes their product on a large fraction of all entries of output on average, then there exists another algorithm that runs in  $\widetilde{O}(T(n))$  time and correctly computes their product on all entries of output. The proof essentially uses the self-correction of linearity [6].

► **Theorem 2.** Fix a finite field  $\mathbb{F}$ . Let  $\alpha \in (0, 1/8)$ . Let ALG be an algorithm that gets as input two matrices  $A, B \in \mathbb{F}^{n \times n}$ , runs in time  $T(n)$ , and outputs a matrix  $\text{ALG}(A, B) \in \mathbb{F}^{n \times n}$ . Suppose that

$$\mathbb{E}_{A, B \in \mathbb{F}^{n \times n}}[\text{agr}(\text{ALG}(A, B), A \cdot B)] > 1 - \alpha .$$

Then, there is an algorithm  $\text{ALG}^*$  that gets as input two matrices  $A, B \in \mathbb{F}^{n \times n}$ , runs in time  $O(T(n) \cdot \log(n))$ , and outputs a matrix  $\text{ALG}^*(A, B) \in \mathbb{F}^{n \times n}$  such that for all  $A, B$  it holds that

$$\Pr[\text{ALG}^*(A, B) = A \cdot B] > 1 - 1/n ,$$

where the randomness is only over the internal coins of  $\text{ALG}^*$ .

**Proof.** Given the algorithm ALG as in the assumption of the theorem, we design  $\text{ALG}^*$  as follows.

■ **Algorithm 1** Approximation for High Agreement Matrix Multiplication Algorithms.

---

**Input:**  $A, B \in \mathbb{F}^{n \times n}$ , ALG

**Output:**  $A \cdot B$

- 1 Let  $k = O(\log(n))$
  - 2 **for**  $r = 0$  to  $k$  **do**
  - 3     Generate two random matrices  $R, S \in \mathbb{F}^{n \times n}$
  - 4     Select two random variables  $\pi, \sigma \in [n]$  independently
  - 5      $M = \text{ALG}((A + R)^{\pi, 0}, (B + S)^{0, \sigma}) - \text{ALG}(R^{\pi, 0}, (B + S)^{0, \sigma}) - \text{ALG}((A + R)^{\pi, 0}, S^{0, \sigma}) + \text{ALG}(R^{\pi, 0}, S^{0, \sigma})$
  - 6     Let  $C_r = M^{n - \pi, n - \sigma}$
  - 7 Define the matrix  $C \in \mathbb{F}^{n \times n}$  by taking the majority vote of all  $C_r$  in each coordinate.
- 

#### Correctness

Consider an entry  $(i, j)$  in the output matrix  $A \cdot B$ . In each iteration we call ALG four times, and in each of the calls the input is distributed uniformly in  $\mathbb{F}^{n \times n}$ . Furthermore, since  $\pi, \sigma$  are chosen uniformly, it follows that  $(i + \pi, j + \sigma)$  are distributed uniformly. Therefore, the probability that all the four calls of ALG produce the correct answer in this entry is at least  $1 - 4\alpha$ . Therefore, for each repetition  $r$ , we have

$$\Pr[(C_r)_{i, j} = (A \cdot B)_{i, j}] \geq 1 - 4\alpha .$$

## 34:8 Matrix Multiplication Reductions

By Chernoff bound, the probability that the majority vote of the  $k$  repetition will produce an incorrect answer is upper bounded by

$$\Pr[(C_r)_{i,j} = (A \cdot B)_{i,j}] \geq \exp(-\Omega((1 - 4\alpha - 1/2) \cdot k)) < 1/n^3 ,$$

Here we make the assumption that  $\alpha$  is bounded below  $1/8$ .

Hence, the probability that a particular entry  $(i, j)$  is incorrect after  $k$  iterations is at most  $n^{-3}$ . By union bound over all entries, the probability that at least one entry is incorrect in the output matrix is at most  $n^2 \cdot n^{-3} = 1/n$ .

### Running time

The total running time is dominated by  $O(\log(n))$  invocations of ALG, and hence, the runtime of  $\text{ALG}^*$  is  $O(T(n) \cdot \log(n))$ . ◀

## 4 Low Agreement with One-Sided Error

In this section we prove Theorem 3. We restate the theorem here for convenience.

► **Theorem 3.** *Let ALG be an algorithm that gets input two matrices  $A, B \in \mathbb{F}_2^{n \times n}$ , runs in time  $T(n)$ , and outputs a matrix  $\text{ALG}(A, B) \in \mathbb{F}_2^{n \times n}$ . Let  $\delta > 0$ , and suppose that*

- $\mathbb{E}_{A, B \in \mathbb{F}_2^{n \times n}} [\text{agr}(\text{ALG}(A, B), A \cdot B)] \geq 1/2 + \delta$ .
- If  $(AB)_{i,j} = 0$ , then  $\text{ALG}(A, B)_{i,j} = 0$ .

*Then, there is an algorithm  $\text{ALG}^*$  that gets as input two matrices  $A, B \in \mathbb{F}_2^{n \times n}$ , runs in time  $\tilde{O}(T(n))$ , and outputs a matrix  $\text{ALG}^*(A, B) \in \mathbb{F}_2^{n \times n}$  such that for all  $A, B$  it holds that*

$$\Pr[\text{ALG}^*(A, B) = A \cdot B] > 1 - 1/n,$$

where the randomness is only over the internal coins of  $\text{ALG}^*$ .

Before proving the theorem, we need some definitions. We start by defining the notion of a *good* coordinate. We say a coordinate  $(i, j) \in [n] \times [n]$  is *good*, if ALG returns 1 at the entry  $(i, j)$  for more than  $\delta/2$  fraction of possible inputs.

► **Definition 12.** *Denote by  $G$  the set of good coordinates, defined as*

$$G = \{(i, j) \in [n] \times [n] : \Pr_{A, B \in \mathbb{F}_2^{n \times n}} [\text{ALG}(A, B)_{i,j} = 1] > \delta/2\} .$$

The following claim is immediate from the definition and the assumptions of the theorem.

▷ **Claim 13.**  $|G| \geq (\delta/2) \cdot n^2$ .

Proof. Let  $p_{i,j} = \Pr_{A, B \in \mathbb{F}_2^{n \times n}} [\text{ALG}(A, B)_{i,j} = 1]$ . Note that by the assumptions of Theorem 3, we have  $\mathbb{E}_{i,j} [p_{i,j}] \geq \delta$ . Note that

$$\delta \leq \mathbb{E}_{i,j \in [n] \times [n]} [p_{i,j}] \leq \Pr_{i,j} [(i, j) \in G] \cdot 1 + \Pr_{i,j} [(i, j) \notin G] \cdot (\delta/2) \leq \Pr_{i,j} [(i, j) \in G] \cdot 1 + 1 \cdot (\delta/2) ,$$

and hence  $\Pr[(i, j) \in G] \geq \delta/2$ , as required. ◀

Next, we define the notion of good input matrices with respect to a good coordinate.



► **Definition 14.** For a coordinate  $(i, j)$ , define  $X_{i,j}$  as follows.

$$X_{i,j} = \{A : \Pr_B[\text{ALG}(A, B)_{i,j} = 1] \geq \delta/4\} .$$

Given a coordinate  $(i, j)$  and a matrix  $A$ , define  $Y_{i,j}^A$  to be the set of matrices  $B$  for which  $\text{ALG}$  returns 1 at the entry  $(i, j)$ . That is,

$$Y_{i,j}^A = \{B : \text{ALG}(A, B)_{i,j} = 1\} .$$

We make the following claim about the densities of  $X_{i,j}$  and  $Y_{i,j}^A$ .

▷ **Claim 15.** For any  $(i, j) \in G$  it holds that  $\Pr_{A \in \mathbb{F}^{n \times n}}[A \in X_{i,j}] \geq \delta/4$ . Furthermore, if  $A \in X_{i,j}$  then  $\Pr_{B \in \mathbb{F}^{n \times n}}[B \in Y_{i,j}^A] \geq \delta/4$ .

*Proof.* Fix a good coordinate  $(i, j) \in G$ , and for each  $A \in \mathbb{F}^{n \times n}$ , let  $p_A = \Pr_{B \in \mathbb{F}^{n \times n}}[\text{ALG}(A, B)_{(i,j)} = 1]$ . From the definition of  $G$  we have  $\mathbb{E}_A[p_A] \geq \delta/2$ .

$$\delta/2 \leq \mathbb{E}_A[p_A] = \Pr_{A \in \mathbb{F}^{n \times n}}[A \in X_{i,j}] \cdot 1 + \Pr_{A \in \mathbb{F}^{n \times n}}[A \notin X_{i,j}] \cdot \delta/4 \leq \Pr_{A \in \mathbb{F}^{n \times n}}[A \in X_{i,j}] + \delta/4 ,$$

and hence,  $\Pr[A \in X_{i,j}] \geq \delta/4$ .

The furthermore part is by definition of  $X_{i,j}$ . ◁

► **Definition 16.** Denote by  $L^k \in \mathbb{F}^{n \times n}$  a random matrix of rank at most  $k$ , constructed by sampling the first  $k$  columns independently uniformly at random from  $\mathbb{F}^n$ , and then taking the remaining  $n - k$  columns to be uniformly random linear combinations of the first  $k$  vectors.

The following lemma is from [2]. It shows that if  $L_A^{2k}$  is a random matrix of rank at most  $2k$  sampled as in Definition 16, then  $M_A = A - (L_A^{2k})$  belongs to any subspace of matrices of co-dimension  $k$  with a non-negligible probability. We provide the proof of the lemma here for completeness.

► **Lemma 17** (Lemma 4.8 from [2]). Fix a matrix  $A \in \mathbb{F}^{n \times n}$ , let  $k$  be a parameter, and let  $\ell \geq 2k$ . Let  $L_A^\ell$  be a random matrix of rank at most  $\ell$  sampled as in Definition 16, and let  $M_A = A - (L_A^\ell)$ . Then, for any subspace  $V \subseteq \mathbb{F}^{n \times n}$  of  $\dim(V) \geq n^2 - k$  it holds that

$$\Pr[M_A \in V] \geq \frac{1}{2|\mathbb{F}|^k} .$$

**Proof.** Since  $V$  has co-dimension at most  $k$ , a matrix in  $V$  must be orthogonal to all the basis vectors of its orthogonal complement. Since there are up to  $k$  such basis vectors, the membership condition of  $M_A$  in  $V$  can be written down in the form of  $k$  linear constraints. Viewing  $M_A$  as a vector in  $\mathbb{F}^{n^2}$ , we can write the  $k$  linear constraints on the elements of the matrix  $M_A^{2k}$  in the form

$$\alpha_1 \cdot (M_A)_{i_1, j_1} + \alpha_2 \cdot (M_A)_{i_2, j_2} + \dots + \alpha_r (M_A)_{i_t, j_t} = 0 .$$

Here,  $\alpha_i$ 's are constants and  $t \in [n^2]$  is the number of elements upon which the constraints depend. Writing  $M_A$  as  $m \in \mathbb{F}^{n^2}$ , we can represent these linear constraints as a system of equations of the form  $G \cdot m = 0$ , where  $G$  is a  $k \times n^2$  matrix. Now, we perform Gaussian elimination on  $G$ , which gives us a matrix  $G'$ , where each row has a 1 entry such that all the other entries in the column containing this 1 are 0. We refer to such 1's as *leading 1's*. That is, by permuting the columns of  $G'$ , we may think of it as being of the form  $G' = [I_k | G^*]$ .

## 34:10 Matrix Multiplication Reductions

Consider the set of  $k$  coordinates of  $m$  corresponding to the  $k$  leading 1s in  $G'$ , one from each row. These  $k$  coordinates of  $m$  in turn correspond to  $k$  pairs of coordinates  $\{(i_1, j_1), (i_2, j_2) \dots (i_k, j_k)\}$  in  $M_A$ . These  $k$  pairs of coordinates in  $M_A$  can belong to at most  $k$  rows in  $M_A$ . We now bound the probability of none of these  $k$  rows in  $L_A^\ell$  being a linear combination of the other rows. Let us denote this event as  $\Omega$ . Then

$$\begin{aligned} \Pr[\Omega] &= \left(1 - \frac{1}{2^\ell}\right) \left(1 - \frac{2}{2^\ell}\right) \left(1 - \frac{4}{2^\ell}\right) \cdots \left(1 - \frac{2^{k-1}}{2^\ell}\right) \\ &\geq \left(1 - \frac{2^{k-1}}{2^\ell}\right)^k \\ &\geq \left(1 - \frac{1}{2^{k+1}}\right)^k \\ &\geq 1 - \frac{k}{2^{k+1}} \geq \frac{1}{2} \end{aligned}$$

If  $\Omega$  happens, then we get a coordinate  $(i_r, j_r)$  in  $M_A$  corresponding to the  $r^{\text{th}}$  linear constraint, for all  $r \in [k]$ , such that no other constraint depends upon it (as it corresponds to a leading 1) and it is chosen uniformly at random (since the rows containing these coordinates are linearly independent). Therefore, the probability that this random value satisfies the  $i^{\text{th}}$  constraint is  $1/|\mathbb{F}|$ . To see this, assume that the values of all other coordinates involved in the  $i^{\text{th}}$  constraint are fixed, then we are left with only one choice for the value of the coordinate  $(c_i, c'_i)$  which satisfies the constraint. Therefore, we have

$$\begin{aligned} \Pr[M_A \in V] &= \Pr[\text{All } k \text{ linear constraints are satisfied}] \\ &= \Pr[\Omega] \cdot \frac{1}{|\mathbb{F}|^k} \geq \frac{1}{2|\mathbb{F}|^k} . \end{aligned}$$

This completes the proof of Lemma 17. ◀

### 4.1 Computing the good coordinates

Next, we start describing the reduction guaranteed by Theorem 3. As a first step we design Algorithm 2, that gets two matrices  $A, B$  and outputs a matrix  $C$  with values in  $\mathbb{F}_2 \cup \{\perp\}$ , satisfying the following guarantees.

1. If  $C_{i,j} \neq \perp$ , then  $C_{i,j}$  contains the correct values, i.e.,  $C_{i^*,j^*} = (A \cdot B)_{i^*,j^*}$ .
2. For any good coordinate  $(i^*, j^*) \in G$  we have  $\Pr[C_{i^*,j^*} = (A \cdot B)_{i^*,j^*}] \geq \delta_0$ , where  $\delta_0$  is some constant that depends only on  $\delta$ . That is, with non-negligible probability  $C_{i^*,j^*}$  contains the correct answer, and not  $\perp$ .

Then, in Section 4.2 we use Algorithm 2 as a subroutine, in order to compute the entire matrix  $A \cdot B$  correctly.

In lines 3-4 we decompose  $A = L_A^{2k} + M_A$ , and  $B = L_B^{2tk} + M_B$  with the intention of computing  $A \cdot B$  by writing

$$\begin{aligned} AB &= (M_A + L_A^{2k}) \cdot (M_B + L_B^{2tk}) \\ &= M_A \cdot M_B + M_A \cdot L_B^{2tk} + L_A^{2k} \cdot M_B + L_A^{2k} \cdot L_B^{2tk} . \end{aligned}$$

Lines 5-13 try to compute  $C = M_A \cdot M_B$ . Then, in line 14, we sum up the 4 terms. Using the fact that multiplication of matrices of rank  $k$  takes  $O(kn^2)$  time, the last three terms can be computed in  $O(tkn^2)$  time, and hence, it remains to compute  $M_A \cdot M_B$ . The remainder of this subsection is dedicated to analyzing lines 5-13, which contain the most involved part of the algorithm.

■ **Algorithm 2** Approximating good coordinates for one-sided error algorithms.

---

**Input:**  $A, B \in \mathbb{F}_2^{n \times n}$ , ALG  
**Output:** An  $n \times n$  matrix  $C$  with values  $\mathbb{F}_2 \cup \{\perp\}$

- 1 Let  $t > \log(4/\delta) + 2$
- 2 Let  $k = O(\log(1/\delta))$  from the “in particular” part of Lemma 8 with  $\alpha = \delta/4$  and  $t$  chosen above
- 3 Sample two random matrices  $L_A^{2k}$  and  $L_B^{2tk}$  of rank at most  $2k$  and  $2tk$  respectively, as in Definition 16
- 4 Define  $M_A = A - L_A^{2k}$  and  $M_B = B - L_B^{2tk}$
- 5 Let  $C$  be the  $n \times n$  matrix initialized with all  $\perp$
- 6 Sample  $t - 1$  random matrices  $R_1, R_2, \dots, R_{t-1} \in \mathbb{F}_2^{n \times n}$  and set  $R_t = M_A - (R_1 + R_2 + \dots + R_{t-1})$
- 7 **for**  $r = 1, \dots, t$  **do**
- 8     Sample  $t - 1$  random matrices  $S_1^{(r)}, S_2^{(r)}, \dots, S_{t-1}^{(r)} \in \mathbb{F}_2^{n \times n}$  and set  $S_t^{(r)} = M_B - (S_1^{(r)} + S_2^{(r)} + \dots + S_{t-1}^{(r)})$
- 9     **for**  $s = 1, \dots, t$  **do**
- 10         Compute  $\text{ALG}(R_r, S_s^{(r)})$
- 11     **for**  $(i, j) \in [n] \times [n]$  **do**
- 12         **if**  $\text{ALG}(R_r, S_s^{(r)})_{i,j} = 1$  for all  $r, s \in \{1, \dots, t\}$  **then**
- 13             Set  $C_{i,j} = \sum_{r,s} \text{ALG}(R_r, S_s^{(r)})_{i,j} \pmod{2}$
- 14 **return**  $C + M_A \cdot L_B^{2tk} + L_A^{2k} \cdot M_B + L_A^{2k} \cdot L_B^{2tk}$  // if  $C_{i,j} = \perp$ , then we return  $\perp$  in the coordinate  $(i, j)$

---

We would like to compute  $M_A \cdot M_B$  by writing  $M_A = R_1 + R_2 + \dots + R_t$ , and  $M_B = S_1^{(r)} + S_2^{(r)} + \dots + S_t^{(r)}$  for  $r = 1 \dots t$ , and then computing  $\text{ALG}(R_r, S_s^{(r)})$  for all  $r, s$ . Note that if we could guarantee that  $\text{ALG}(R_r, S_s^{(r)})$  returns  $R_r \cdot S_s^{(r)}$ , then, we would have

$$M_A \cdot M_B = \sum_{r,s} R_r \cdot S_s^{(r)} = \sum_{r,s} \text{ALG}(R_r, S_s^{(r)}) . \quad (1)$$

However, ALG is not guaranteed to return the product of the inputs correctly. Instead, we claim that (1) for *some* good coordinates  $(i^*, j^*)$  it holds  $C_{i^*, j^*} = (M_A \cdot M_B)_{i^*, j^*}$ , and (2) the remaining coordinates in  $C$  remain  $\perp$ . This is summarized formally in the next two claims.

▷ **Claim 18.** For any  $(i, j) \in [n] \times [n]$  if  $C_{i,j} \in \{0, 1\}$  (i.e.,  $C_{i,j} \neq \perp$ ), then  $C_{i,j} = (M_A \cdot M_B)_{i,j}$ .

Proof. Fix any coordinate  $(i, j)$ . Note that in line 13 we set  $C_{i,j} = \sum_{r,s} \text{ALG}(R_r, S_s^{(r)})_{i,j} \pmod{2}$  only if  $\text{ALG}(R_r, S_s^{(r)})_{i,j} = 1$  for all  $r, s$ . Recall that by the assumption of the algorithm if  $\text{ALG}(R_r, S_s^{(r)})_{i,j} = 1$ , then  $\text{ALG}(R_r, S_s^{(r)})_{i,j} = (R_r \cdot S_s^{(r)})_{i,j}$ . The claim follows by Equation (1) restricted to the coordinate  $(i, j)$ , as

$$C_{i,j} = \sum_{r,s} \text{ALG}(R_r, S_s^{(r)})_{i,j} = \sum_{r,s} (R_r \cdot S_s^{(r)})_{i,j} = (M_A \cdot M_B)_{i,j} ,$$

as required. ◁

▷ **Claim 19.** Fix a good coordinate  $(i^*, j^*) \in G$ . Then  $\Pr[C_{i^*, j^*} = (M_A \cdot M_B)_{i^*, j^*}] \geq \delta_0 = 0.5^{O(\log^3(1/\delta))}$ .

## 34:12 Matrix Multiplication Reductions

Proof. Consider the set  $X_{i^*,j^*}$  from Definition 14 for a good entry  $(i^*, j^*) \in G$ . By Claim 15, the density of  $X_{i,j}$  is at least  $\delta/4$ , and hence, Lemma 8 guarantees the existence of an affine subspace  $V_{i^*,j^*}$  of dimension  $\dim(V_{i^*,j^*}) \geq n - k$ . Then, using Lemma 17 with  $V_{i^*,j^*}$  we have

$$\Pr[M_A \in V_{X_{i,j}}] \geq \frac{1}{2 \cdot 2^k} . \quad (2)$$

Let us condition on the event that  $M^A \in V_{X_{i^*,j^*}}$ . Then by Lemma 8,

$$\Pr_{\substack{R_1, R_2, \dots, R_t \in \mathbb{F}_n^{n \times n} \\ \sum_r R_r = M_A}} [R_1, R_2, \dots, R_t \in X_{i,j}] \geq (\delta/8)^t . \quad (3)$$

For each of  $R_1, \dots, R_t$  define the sets  $Y^{R_1}, Y^{R_2}, \dots, Y^{R_t}$  as in Definition 14. (Recall  $Y^R$  is the set of all matrices  $S$  such that  $(R \cdot S)_{i^*,j^*} = 1$ . We omit the subscript  $(i^*, j^*)$  for readability.)

From Claim 15, we know each of  $Y^{R_1}, \dots, Y^{R_t}$  has density at least  $\delta/4$ . Hence, by applying Lemma 8 on each of them, we obtain subspaces  $V_{Y^{R_1}}, \dots, V_{Y^{R_t}}$  of co-dimension at most  $k$ . Define  $V_Y = V_{Y^{R_1}} \cap V_{Y^{R_2}} \cap \dots \cap V_{Y^{R_t}}$  to be their intersection, and note that  $\dim(V_Y) \geq n - tk$ . Therefore, by applying Lemma 17 on the matrix  $B$  with the subspace  $V_Y$ , we get <sup>1</sup>

$$\Pr[M_B \in V_Y] \geq \frac{1}{2 \cdot 2^{tk}} . \quad (4)$$

Conditioning further on the event that  $M_B \in V_Y$ , we apply Lemma 8, and for each  $r = 1, \dots, t$  we get

$$\Pr_{\substack{S_1^{(r)}, \dots, S_t^{(r)} \\ \sum_s S_s^{(r)} = M_B}} [S_1^{(r)}, \dots, S_t^{(r)} \in Y^{R_r}] \geq (\delta/8)^t .$$

Since the events above are independent between different  $r$ 's, the probability that the algorithm returns correct output on the entry  $(i^*, j^*)$  is lower bounded by the product of the probabilities in Equations (2)–(4), and hence

$$\Pr[C_{i^*,j^*} = (M_A \cdot M_B)_{i^*,j^*}] \geq \frac{1}{2^{k+1}} \times (\delta/8)^t \times \frac{1}{2 \cdot 2^{tk}} \times ((\delta/8)^t)^t \geq \frac{1}{2^{O(\log^3(1/\delta))}} .$$

This completes the proof of Claim 19.  $\triangleleft$

## 4.2 Proof of Theorem 3

We are now ready to prove Theorem 3. Algorithm 3 uses Algorithm 2 as a subroutine, by running it several times. We claim that Algorithm 3 correctly computes the correct answer with high probability for any input  $A, B$ . Note that Algorithm 2 is guaranteed to be correct only for good coordinates, although it does not get the good coordinates as an input, and the guarantee about the good coordinates only appears in the analysis.

<sup>1</sup> Note that although the algorithm samples  $M_B$  before  $R_1, \dots, R_t$ , in fact they are sampled independently of each other, and hence Lemma 17 is applicable here.

■ **Algorithm 3** Approximation for one-sided Agreement Matrix Multiplication Algorithms.

---

**Input:**  $A, B \in \mathbb{F}_2^{n \times n}$   
**Output:**  $A \cdot B$

- 1 Let  $C$  be the  $n \times n$  matrix initialized with all  $\perp$ .
- 2 Let  $\delta_0$  be the constant from Claim 19
- 3 **repeat**  $O\left(\frac{\log(n)}{\delta \cdot \delta_0}\right)$  **times**
- 4     Sample uniformly random  $\pi, \sigma \in [n]$ .
- 5     Run Algorithm 2 with the inputs as  $A^{\pi,0}, B^{0,\sigma}$ , ALG.
- 6     Let  $C^*$  be the resulting matrix
- 7     **for**  $(i, j) \in [n] \times [n]$  **do**
- 8         **if**  $C_{i+\pi \pmod n, j+\sigma \pmod n}^* \neq \perp$  **then**
- 9             Set  $C_{i,j} = C_{i+\pi, j+\sigma}^*$
- 10 **return**  $C$

---

The following claim completes the proof of Theorem 3.

▷ **Claim 20.** Fix a coordinate  $(i, j) \in [n] \times [n]$ . Algorithm 3 returns the matrix  $C$  such that  $\Pr[C_{i,j} = (A \cdot B)_{i,j}] \geq 1 - 1/n^3$ .

In particular, by taking the union bound over all coordinates  $(i, j)$  it follows that for any input  $A, B$  Algorithm 3 returns their product with probability at least  $1 - 1/n$ .

*Proof.* Fix a coordinate  $(i, j) \in [n] \times [n]$ . The algorithm chooses random  $\pi$  and  $\sigma$ , and runs Algorithm 2 on the shifted matrices  $A^{\pi,0}$  and  $B^{0,\sigma}$ .

Note that since  $\pi$  and  $\sigma$  are chosen uniformly at random, it follows that  $\Pr[(i + \pi \pmod n, j + \sigma \pmod n) \in G] = |G|/n^2 \geq \delta/2$ .

Suppose that  $(i + \pi \pmod n, j + \sigma \pmod n)$  is indeed a good coordinate. Then by Claim 20 with probability  $\delta_0$  we obtain the correct answer in the coordinate  $(i + \pi \pmod n, j + \sigma \pmod n)$ , in which case we set  $C_{i,j}$  to be that answer  $(A \cdot B)_{i,j}$ . Otherwise,  $C_{i,j}$  remains  $\perp$ .

Therefore, with probability at least  $(\delta/2) \cdot \delta_0$  in each iteration  $C_{i,j}$  changes from  $\perp$  to  $(A \cdot B)_{i,j}$ , and once it changes, it never changes its value again.

By repeating the procedure  $R = O\left(\frac{\log(n)}{\delta \cdot \delta_0}\right)$  times, the probability that in the end of the algorithm  $C_{i,j} = \perp$  is upper bounded by  $\Pr[C_{i,j} = \perp] \leq (1 - \delta_0)^R < 1/n^3$ . This completes the proof of the claim. ◁

We conclude the proof with the analysis of the running time of the algorithm.

### Running Time

The total running time of Algorithm 3 is essentially dominated by the running time of Algorithm 2 multiplied by  $O\left(\frac{\log(n)}{\delta \cdot \delta_0}\right)$ . Each iteration of Algorithm 2 involves  $O(t^2)$  calls to ALG plus additional  $O(tn^2)$  time. Therefore, the running time is  $O(t^2 \log(n)T(n)/\delta \cdot \delta_0)$ . Since  $t = O(\log 1/\delta)$ , and  $\delta_0 = 0.5^{O(\log^3(1/\delta))}$  the total running time of the algorithm is  $2^{O(\log^3(1/\delta))} \cdot T(n) \log(n)$ .

In particular, even for a slightly sub-constant  $\delta \geq \exp(-\log^{0.33}(n))$ , our algorithm runs in time  $T(n) \cdot n^{o(1)}$ .

This completes the proof of Theorem 3.

## References

- 1 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA 2021*, pages 522–539. SIAM, 2021.
- 2 Vahid R. Asadi, Alexander Golovnev, Tom Gur, and Igor Shinkar. Worst-case to average-case reductions via additive combinatorics. In *STOC 2022*, pages 00–00. ACM, 2022.
- 3 Vahid R. Asadi, Alexander Golovnev, Tom Gur, Igor Shinkar, and Sathyawageeswar Subramanian. *Quantum Worst-Case to Average-Case Reductions for All Linear Problems*, pages 2535–2567. Society for Industrial and Applied Mathematics, 2024. doi:10.1137/1.9781611977912.90.
- 4 Shai Ben-David, Benny Chor, Oded Goldreich, and Michel Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.
- 5 Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti.  $O(n^{2.7799})$  complexity for  $n \times n$  approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979. doi:10.1016/0020-0190(79)90113-3.
- 6 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *STOC 1990*, pages 73–83. ACM, 1990.
- 7 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1):1–106, October 2006. doi:10.1561/0400000004.
- 8 Mei-Chu Chang. A polynomial bound in Freiman’s theorem. *Duke Mathematical Journal*, 113(3):399–419, 2002. doi:10.1215/S0012-7094-02-11331-3.
- 9 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial. doi:10.1016/S0747-7171(08)80013-2.
- 10 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 2129–2138. IEEE, 2023. doi:10.1109/FOCS57990.2023.00130.
- 11 Leszek Gąsieniec, Christos Levcopoulos, Andrzej Lingas, Rasmus Pagh, and Takeshi Tokuyama. Efficiently correcting matrix products. *Algorithmica*, 79(2):428–443, October 2017. doi:10.1007/s00453-016-0202-3.
- 12 Oded Goldreich, editor. *Studies in complexity and cryptography: miscellanea on the interplay between randomness and computation*. Springer-Verlag, Berlin, Heidelberg, 2011.
- 13 Shuichi Hirahara and Nobutaka Shimizu. Hardness self-amplification: Simplified, optimized, and unified. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 70–83. Association for Computing Machinery, 2023. doi:10.1145/3564246.3585189.
- 14 Russell Impagliazzo. A personal view of average-case complexity. *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, 1995. URL: <https://api.semanticscholar.org/CorpusID:2154064>.
- 15 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 104–114. IEEE Computer Society, 2011. doi:10.1109/CCC.2011.34.
- 16 Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89604.
- 17 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC ’14*, pages 296–303, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2608628.2608664.

- 18 Leonid A. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986. doi:10.1137/0215020.
- 19 V. Ya. Pan. Strassen’s algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations. In *19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, pages 166–176, 1978. doi:10.1109/SFCS.1978.34.
- 20 Francesco Romani. Some properties of disjoint sums of tensors related to matrix multiplication. *SIAM Journal on Computing*, 11(2):263–267, 1982. doi:10.1137/0211020.
- 21 A. Schönhage. Partial and total matrix multiplication. *SIAM Journal on Computing*, 10(3):434–455, 1981. doi:10.1137/0210032.
- 22 Andrew James Stothers. On the complexity of matrix multiplication. In *University of Edinburgh*, 2010. URL: <https://api.semanticscholar.org/CorpusID:262795811>.
- 23 V. Strassen. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 49–54, 1986. doi:10.1109/SFCS.1986.52.
- 24 Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.
- 25 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, STOC ’12*, pages 887–898, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214056.





# Testing Intersectingness of Uniform Families

Ishay Haviv

The Academic College of Tel Aviv-Yaffo, Tel Aviv 61083, Israel

Michal Parnas

The Academic College of Tel Aviv-Yaffo, Tel Aviv 61083, Israel

---

## Abstract

A set family  $\mathcal{F}$  is called intersecting if every two members of  $\mathcal{F}$  intersect, and it is called uniform if all members of  $\mathcal{F}$  share a common size. A uniform family  $\mathcal{F} \subseteq \binom{[n]}{k}$  of  $k$ -subsets of  $[n]$  is  $\varepsilon$ -far from intersecting if one has to remove more than  $\varepsilon \cdot \binom{n}{k}$  of the sets of  $\mathcal{F}$  to make it intersecting. We study the property testing problem that given query access to a uniform family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , asks to distinguish between the case that  $\mathcal{F}$  is intersecting and the case that it is  $\varepsilon$ -far from intersecting. We prove that for every fixed integer  $r$ , the problem admits a non-adaptive two-sided error tester with query complexity  $O(\frac{\ln n}{\varepsilon})$  for  $\varepsilon \geq \Omega((\frac{k}{n})^r)$  and a non-adaptive one-sided error tester with query complexity  $O(\frac{\ln k}{\varepsilon})$  for  $\varepsilon \geq \Omega((\frac{k^2}{n})^r)$ . The query complexities are optimal up to the logarithmic terms. For  $\varepsilon \geq \Omega((\frac{k^2}{n})^2)$ , we further provide a non-adaptive one-sided error tester with optimal query complexity of  $O(\frac{1}{\varepsilon})$ . Our findings show that the query complexity of the problem behaves differently from that of testing intersectingness of non-uniform families, studied recently by Chen, De, Li, Nadimpalli, and Servedio (ITCS, 2024).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Mathematics of computing  $\rightarrow$  Combinatorial algorithms

**Keywords and phrases** Intersecting family, Uniform family, Property testing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.35

**Category** RANDOM

**Related Version** *Full Version*: <http://arxiv.org/abs/2404.11504>

**Funding** *Ishay Haviv*: Research supported in part by the Israel Science Foundation (grant No. 1218/20).

**Acknowledgements** We would like to thank the anonymous reviewers for their useful comments.

## 1 Introduction

A set family  $\mathcal{F}$  is called intersecting if for every two sets  $F_1, F_2 \in \mathcal{F}$ , it holds that  $F_1 \cap F_2 \neq \emptyset$ . The study of intersecting families plays a central role in the area of extremal combinatorics with a particular attention dedicated to the uniform case, where all the sets of the family share a common size. One of the most influential results in this context is the Erdős–Ko–Rado theorem [4], proved in 1938 and published in 1961, which states that for integers  $n$  and  $k$  with  $n \geq 2k$ , the maximum size of an intersecting family of  $k$ -subsets of  $[n] = \{1, 2, \dots, n\}$  is  $\binom{n-1}{k-1}$ , attained by the families of all  $k$ -subsets that include a fixed element. Another prominent result, proved by Lovász [10] in 1978 settling a conjecture of Kneser [9] from 1955, asserts that for  $n \geq 2k$ , the family  $\binom{[n]}{k}$  of all  $k$ -subsets of  $[n]$  cannot be covered by fewer than  $n - 2k + 2$  intersecting families. This result is tight, as follows by considering, for each  $i \in [n - 2k + 1]$ , the family of  $k$ -subsets of  $[n]$  that include  $i$ , and the family of  $k$ -subsets of  $[n] \setminus [n - 2k + 1]$ . A more recent result, proved by Dinur and Friedgut [3] in 2009, provides a structural characterization for large intersecting families of  $k$ -subsets of  $[n]$  when  $k$  is sufficiently smaller than  $n$ . It says, roughly speaking, that every such family is



© Ishay Haviv and Michal Parnas;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 35; pp. 35:1–35:15



Leibniz International Proceedings in Informatics

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

approximately contained in an intersecting junta, that is, an intersecting family  $\mathcal{J}$  over  $[n]$ , such that the membership of a set  $F$  in  $\mathcal{J}$  depends only on  $F \cap J$  for a fixed set  $J \subseteq [n]$ , where the size of  $J$  is determined by the precision of the containment (see Theorem 7).

In this paper, we investigate intersecting uniform families from the computational perspective of property testing. This field delves into the amount of data required for distinguishing objects that satisfy a prescribed property from those that significantly deviate from satisfying it. The objective is to design a randomized algorithm for this task, called a (two-sided error) tester, that succeeds with high constant probability and minimizes the query complexity, i.e., the number of queries to the input object. If the tester accepts objects that satisfy the given property with probability 1, we say that its error is one-sided. The tester is said to be non-adaptive if its queries are determined independently of the answers provided for prior queries. For a thorough introduction to the field of property testing, the reader is referred to, e.g., [8].

For integers  $n$  and  $k$  with  $n \geq 2k$  and for a real  $\varepsilon \in [0, 1)$ , we say that a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting if one has to remove more than  $\varepsilon \cdot \binom{[n]}{k}$  of its sets to make it intersecting. In the property testing problem  $\text{INTERSECTING}_{n,k,\varepsilon}$ , we are given access to a family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , represented by an indicator function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$ , and the goal is to distinguish between the case that  $\mathcal{F}$  is intersecting and the case that it is  $\varepsilon$ -far from intersecting. Note that the Erdős–Ko–Rado theorem [4] implies that every intersecting family  $\mathcal{F} \subseteq \binom{[n]}{k}$  includes at most a  $k/n$  fraction of the sets of  $\binom{[n]}{k}$ . This gives impetus to studying the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem for a proximity parameter  $\varepsilon = \varepsilon(n, k)$  that diminishes faster than the ratio  $k/n$  (see the discussion at the end of Section 3).

Our interest in testing intersectingness of uniform families is sparked and inspired by a recent paper of Chen, De, Li, Nadimpalli, and Servedio [2], who introduced and explored the analogue problem for general (non-uniform) families of sets. In their setting, the input consists of an indicator function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  of a family  $\mathcal{F}$  of subsets of  $[n]$  (of any size), identified with their characteristic vectors in  $\{0, 1\}^n$ , and the goal is to decide whether  $\mathcal{F}$  is intersecting or  $\varepsilon$ -far from intersecting. Here, since the size of the domain is  $2^n$ , a family is said to be  $\varepsilon$ -far from intersecting if more than  $\varepsilon \cdot 2^n$  of its sets should be removed to make it intersecting. Chen et al. [2] proved that this problem admits a non-adaptive one-sided error tester with query complexity  $\text{poly}(n\sqrt{n \cdot \log(1/\varepsilon)}, \frac{1}{\varepsilon})$ . They further established a nearly matching lower bound, showing that the query complexity of every non-adaptive one-sided error tester for the problem is  $2^{\Omega(\sqrt{n \cdot \log(1/\varepsilon)})}$ , whenever  $\varepsilon \in [2^{-n}, \varepsilon_0]$  for some constant  $\varepsilon_0 > 0$ . For non-adaptive two-sided error testers, they obtained a lower bound of  $2^{\Omega(n^{1/4}/\sqrt{\varepsilon})}$  on the query complexity, assuming that  $\varepsilon \in [1/\sqrt{n}, \varepsilon_0]$  for some constant  $\varepsilon_0 > 0$ . As will be shortly described, the results of the present paper highlight a significant difference between the behavior of the query complexity of testing intersectingness in the uniform and non-uniform settings.

## 1.1 Our Contribution

This paper studies the query complexity of the  $\text{INTERSECTING}_{n,k,\varepsilon}$  testing problem. We offer nearly matching upper and lower bounds for various settings of the integers  $n$  and  $k$  and the proximity parameter  $\varepsilon = \varepsilon(n, k)$ . Let us mention already here that all of our upper bounds are proved via efficient testers, whose running time is polynomial in  $n$ . We further note that the results presented below are applicable for all integers  $n$  and  $k$  for which the conditions on  $\varepsilon$  allow it to be smaller than 1. The precise and formal statements are given in the upcoming technical sections.

Our first result furnishes a non-adaptive two-sided error tester for the case of  $\varepsilon \geq \Omega((k/n)^r)$  for a fixed constant  $r$ . Its analysis borrows the structural characterization of large intersecting uniform families due to Dinur and Friedgut [3]. Note that the multiplicative factors hidden by the  $O(\cdot)$  and  $\Omega(\cdot)$  notations might depend on  $r$ .

► **Theorem 1 (Two-Sided Error Tester).** *For every fixed integer  $r$ , for all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \geq \Omega((\frac{k}{n})^r)$ , there exists a non-adaptive two-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  with  $O(\frac{\ln n}{\varepsilon})$  queries.*

In fact, we prove Theorem 1 in a stronger form, adopting the concept of tolerant property testing, introduced by Parnas, Ron, and Rubinfeld [12]. Strengthening the standard notion of property testing, a tolerant tester is required to accept not only objects that satisfy the given property, but also those that are close to satisfying it (and as usual, to reject objects that significantly deviate from the property). Accordingly, the tester given in Theorem 1 is shown to accept with high probability any family that can be made intersecting by removing relatively few of its sets (see Theorem 6). Note that a result of Tell [13] implies that a two-sided error is essentially inherent in tolerant property testing.

We next turn our attention to designing one-sided error testers for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem, wherein an intersecting family must be accepted with probability 1. A natural non-adaptive tester for this purpose, termed the canonical tester, selects  $m$  random sets from  $\binom{[n]}{k}$ , uniformly and independently, and checks whether they include two sets that demonstrate the non-intersectingness of the input family, namely, two disjoint sets within the family. This raises the combinatorial question, which might be of independent interest, of determining the smallest number  $m = m(n, k, \varepsilon)$  of random sets from  $\binom{[n]}{k}$  that guarantee with high probability a pair of disjoint sets that lie in a family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , assuming that  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting. As our main technical contribution, we address this question for the case where  $\varepsilon \geq \Omega((k^2/n)^r)$  for a fixed constant  $r$ . Our analysis yields the following result.

► **Theorem 2 (One-Sided Error Tester).** *For every fixed integer  $r$ , for all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \geq \Omega((\frac{k^2}{n})^r)$ , there exists a non-adaptive one-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  with  $O(\frac{\ln k}{\varepsilon})$  queries.*

Let us emphasize that the tester provided by Theorem 2 surpasses that of Theorem 1 in two respects: its error is one-sided, and its query complexity is lower, replacing the  $\ln n$  term by  $\ln k$ . On the other hand, Theorem 2 requires  $\varepsilon$  to satisfy  $\varepsilon \geq \Omega((k^2/n)^r)$ , and is thus applicable only for  $n \geq \Omega(k^2)$ , whereas the two-sided error tester of Theorem 1 is applicable already for  $n \geq \Omega(k)$ .

For the special case of  $r = 2$ , we offer a notably simple analysis of the canonical tester, enabling us to enhance the query complexity achieved in Theorem 2 by getting rid of the logarithmic term. This gives the following result.

► **Theorem 3 (One-Sided Error Tester;  $r = 2$ ).** *For all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \geq \Omega((\frac{k^2}{n})^2)$ , there exists a non-adaptive one-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  with  $O(\frac{1}{\varepsilon})$  queries.*

We further consider the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem for integers  $n$  and  $k$  satisfying  $n = \alpha \cdot k$  for an arbitrary constant  $\alpha \geq 2$ . Interestingly, the canonical tester fails in this case, because its random samples are unlikely to include even a single pair of disjoint sets, unless the number of samples is exponential in  $n$ . Nevertheless, we show that for all constants  $\alpha \geq 2$  and  $\varepsilon \in (0, 1)$ , the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem with  $n = \alpha \cdot k$  admits a non-adaptive one-sided error tester with constant query complexity. The proof employs a result of Friedgut and Regev [7], and the details are given in Section 4.2 (see Theorem 17).

Our final result supplies a lower bound on the query complexity of the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem. Note that the lower bound applies even to adaptive two-sided error testers. While the result can be derived from a general result of [5], our proof in the full version of the paper explicitly presents hard instances of the problem.

► **Theorem 4 (Lower Bound).** *For all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon = \varepsilon(n, k)$  with  $\binom{n}{k}^{-1} \leq \varepsilon < \frac{1}{2}$ , the query complexity of every tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  is  $\Omega(\frac{1}{\varepsilon})$ .*

It is noteworthy that Theorem 4 implies that the query complexities achieved by our testers for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem are nearly tight. Specifically, the query complexity obtained in Theorem 3 is tight up to a multiplicative constant, while those of Theorems 1 and 2 are tight up to multiplicative logarithmic terms. An intriguing task for further research would be to decide whether these logarithmic terms can be avoided. More ambitiously, it would be interesting to determine the query complexity of the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem for general values of  $n$ ,  $k$ , and  $\varepsilon = \varepsilon(n, k)$ .

## 1.2 Proof Techniques

We provide here a high-level description of the ideas applied in the proofs of Theorems 1 and 2. Let us start with the proof of Theorem 1, which gives a non-adaptive two-sided error tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem with query complexity  $O(\frac{\ln n}{\varepsilon})$ , where  $\varepsilon \geq \Omega(\frac{k}{n})^r$  for a fixed integer  $r$ . Given access to a family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , our tester attempts to decide whether  $\mathcal{F}$  is approximately contained in an intersecting  $j$ -junta over  $[n]$  for some integer  $j$  that depends solely on  $r$ . To do so, the tester picks random sets from  $\binom{[n]}{k}$ , uniformly and independently, and checks whether they lie in  $\mathcal{F}$ . These samples are used to estimate, for each intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , the fraction of sets in  $\binom{[n]}{k}$  that lie in  $\mathcal{F} \setminus \mathcal{J}$ . Our analysis shows that  $O(\frac{\ln n}{\varepsilon})$  samples suffice for these estimations to be pretty accurate with high probability. Then, if those estimations indicate that  $\mathcal{F}$  is approximately contained in some intersecting  $j$ -junta over  $[n]$ , the tester predicts that  $\mathcal{F}$  is close to intersecting. Otherwise, relying on the aforementioned structural result of Dinur and Friedgut [3], the tester deduces that  $\mathcal{F}$  is far from intersecting with high probability. Let us stress that the tester is two-sided error, because even if  $\mathcal{F}$  is intersecting and is essentially aligned with some intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , the random samples might wrongly suggest, with low probability, that  $\mathcal{F}$  significantly deviates from  $\mathcal{J}$ . As previously noted, Theorem 1 is proved with respect to tolerant property testing. For the precise statement and argument, the reader is referred to Section 3.

We next outline the approach applied in the proof of Theorem 2, which establishes a non-adaptive one-sided error tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem with query complexity  $O(\frac{\ln k}{\varepsilon})$ , where  $\varepsilon \geq \Omega(\frac{k^2}{n})^r$  for a fixed integer  $r$ . As mentioned earlier, the proof of this result relies on the canonical tester, which selects random sets from  $\binom{[n]}{k}$ , uniformly and independently, and checks if they include two sets that violate the intersectingness of the given family. Therefore, our goal is to show that if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting, then a collection of  $O(\frac{\ln k}{\varepsilon})$  random sets from  $\binom{[n]}{k}$  includes with high probability two disjoint sets within  $\mathcal{F}$ . Consider a family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , and assume that  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting. Notice that this in particular implies that  $|\mathcal{F}| > \varepsilon \cdot \binom{n}{k}$ , as otherwise, one could remove at most  $\varepsilon \cdot \binom{n}{k}$  of its sets to make it intersecting.

Let us first suppose that the chosen random sets include, for each set  $A \subseteq [n]$  of size at most  $r-1$ , a set  $F_A \in \mathcal{F}$  with  $F_A \cap A = \emptyset$ . A key observation in our approach is that the number of sets in  $\binom{[n]}{k}$  that intersect all of those sets  $F_A$  does not exceed  $k^r \cdot \binom{n-r}{k-r}$ . To see

this, observe that every set that intersects them all includes an element  $j_1 \in F_\emptyset$ , an element  $j_2 \in F_{\{j_1\}}$ , an element  $j_3 \in F_{\{j_1, j_2\}}$ , and so on, up to an element  $j_r \in F_{\{j_1, \dots, j_{r-1}\}}$ , where the  $r$  elements  $j_1, \dots, j_r$  are distinct. It thus follows that there exists a collection of at most  $k^r$  subsets of  $[n]$  of size  $r$ , such that every set in  $\binom{[n]}{k}$  that intersects all the sampled sets that lie in  $\mathcal{F}$  contains at least one of the sets of the collection. This implies that the number of those sets does not exceed  $k^r \cdot \binom{n-r}{k-r} \leq \left(\frac{k^2}{n}\right)^r \cdot \binom{n}{k}$ . Now, by combining our assumption on  $\varepsilon$  with the fact that  $|\mathcal{F}| > \varepsilon \cdot \binom{n}{k}$ , one can show that a random set from  $\binom{[n]}{k}$  lies in  $\mathcal{F}$  and is disjoint from at least one of the previously sampled sets that lie in  $\mathcal{F}$  with non-negligible probability. Therefore, a few additional random sets from  $\binom{[n]}{k}$  are expected to provide with high probability the desired witness for the non-intersectingness of  $\mathcal{F}$ .

The scenario discussed above, however, is somewhat oversimplified. It definitely might be the case that for some set  $A \subseteq [n]$  of size at most  $r - 1$ , the sets of  $\mathcal{F}$  that are disjoint from  $A$  are quite rare, and as such, our random samples are not expected to include them. Following the terminology of [3], we say that such a set  $A$  captures  $\mathcal{F}$  (see Definition 13). To deal with this situation, we show that if a set  $A$  of size at most  $r - 1$  captures  $\mathcal{F}$ , then it admits two disjoint subsets  $B, C \subseteq A$ , such that the sub-families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  of  $\mathcal{F}$ , defined as the restrictions of  $\mathcal{F}$  to the sets whose intersection with  $A$  is, respectively,  $B$  and  $C$ , are far from being cross-intersecting, that is, one has to remove plenty of sets from  $\mathcal{F}_1$  and  $\mathcal{F}_2$  so that every set of  $\mathcal{F}_1$  will intersect every set of  $\mathcal{F}_2$ . Since  $B$  and  $C$  are disjoint, this essentially allows us to ignore from now on the elements of  $A$ , and to analyze the probability that the random samples include two disjoint sets, one from  $\mathcal{F}_1$  and one from  $\mathcal{F}_2$ . By a slight variant of the key observation described above, we wish our samples to include, for each set  $A' \subseteq [n] \setminus A$  of size at most  $r - 1$ , a set  $F_{A'} \in \mathcal{F}_2$  with  $F_{A'} \cap A' = \emptyset$ . If the samples include such sets, then it can be shown that a random set is expected to lie in  $\mathcal{F}_1$  and be disjoint from some previously chosen set of  $\mathcal{F}_2$  with non-negligible probability. However, if some set  $A' \subseteq [n] \setminus A$  of size at most  $r - 1$  captures  $\mathcal{F}_2$ , then a suitable set  $F_{A'}$  is unlikely to be found. In this case, we repeat the above process and further refine  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , fixing the intersections of their sets with  $A'$  to some disjoint subsets and keeping them far from cross-intersecting. It might be needed to repeat this process multiple times, but a crucial component of our argument shows that  $r$  iterations suffice. Indeed, after this number of iterations, it turns out that the refined family  $\mathcal{F}_2$  is already too small to still satisfy the invariant that  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are far from cross-intersecting.

To summarize, our analysis of the canonical tester shows that if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting for a sufficiently large  $\varepsilon = \varepsilon(n, k)$ , then there exist a set  $A \subseteq [n]$  and two disjoint subsets  $B, C \subseteq A$ , such that (a) the restrictions  $\mathcal{F}_1$  and  $\mathcal{F}_2$  of  $\mathcal{F}$  to the sets whose intersections with  $A$  are  $B$  and  $C$ , respectively, are far from cross-intersecting, and (b) no subset of  $[n] \setminus A$  of size at most  $r - 1$  captures  $\mathcal{F}_2$  (see Lemma 15). This allows us to show that  $O\left(\frac{\ln k}{\varepsilon}\right)$  sets chosen randomly from  $\binom{[n]}{k}$  include with high probability a collection of sets of  $\mathcal{F}_2$ , such that relatively few sets of  $\mathcal{F}_1$  intersect them all. Therefore, a few additional samples from  $\binom{[n]}{k}$  are expected to include a set of  $\mathcal{F}_1$  that is disjoint from one of the previously picked sets of  $\mathcal{F}_2$ , resulting in the desired witness for non-intersectingness (see Lemma 16). For the precise and full argument, the reader is referred to Section 4.1.2.

### 1.3 Related Work

The Erdős–Ko–Rado theorem [4] implies that for all integers  $n$  and  $k$  with  $n \geq 2k$ , every family  $\mathcal{F} \subseteq \binom{[n]}{k}$  whose size exceeds  $\binom{n-1}{k-1}$  is not intersecting. One may thus ask whether such a family  $\mathcal{F}$  must include a set that is disjoint from many of the sets of  $\mathcal{F}$ . This question was recently investigated by Frankl and Kupavskii [6] and by Chau, Ellis, Friedgut, and

Lifshitz [1], who provided a positive answer in a strong sense. Namely, it was shown in [1] that for any  $\delta > 0$ , there exists some  $\alpha > 0$ , such that for all integers  $n$  and  $k$  with  $n \geq \alpha \cdot k$ , every family  $\mathcal{F} \subseteq \binom{[n]}{k}$  of size  $|\mathcal{F}| = \binom{n-1}{k-1} + 1$  includes a set that is disjoint from at least  $(\frac{1}{2} - \delta) \cdot \binom{n-k-1}{k-1}$  of the sets of  $\mathcal{F}$ .

We point out that the combinatorial question arising in our analysis of the canonical tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem is similar in spirit to the question studied in [6, 1]. Indeed, the analysis requires us to show that if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting, then a relatively small collection of random sets, chosen uniformly and independently from  $\binom{[n]}{k}$ , includes with high probability two disjoint sets that lie in  $\mathcal{F}$ . For such a statement, one has to show not only the existence of a set in  $\mathcal{F}$  that is disjoint from many of the sets of  $\mathcal{F}$ , but that many of the sets of  $\mathcal{F}$  satisfy this property. Yet, a crucial difference between the two concepts is that in ours, the family  $\mathcal{F}$  is not assumed to exceed the Erdős–Ko–Rado threshold but rather to be  $\varepsilon$ -far from intersecting.

## 1.4 Outline

The rest of the paper is organized as follows. In Section 2, we provide some definitions and tools that will be used throughout the paper. In Section 3, we present and analyze our two-sided error tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem and confirm Theorem 1. Finally, in Section 4, we present and analyze the one-sided error canonical tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem and confirm Theorems 2 and 3. We further present there another one-sided error tester, appropriate for integers  $n$  and  $k$  with  $n = \Theta(k)$ . The proof of Theorem 4 can be found in the full version of the paper.

## 2 Preliminaries

Throughout the paper, we omit all floor and ceiling signs, whenever these are not crucial.

### 2.1 Intersecting Families

For integers  $n$  and  $k$ , let  $\binom{[n]}{k}$  denote the family of all  $k$ -subsets of  $[n] = \{1, \dots, n\}$ . A family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is called intersecting if for every two sets  $F_1, F_2 \in \mathcal{F}$ , it holds that  $F_1 \cap F_2 \neq \emptyset$ . For a real  $\varepsilon \in [0, 1]$ , we say that  $\mathcal{F}$  is  $\varepsilon$ -close to intersecting if it is possible to make  $\mathcal{F}$  intersecting by removing at most  $\varepsilon \cdot \binom{n}{k}$  of its sets. Otherwise, we say that  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting.

For two families  $\mathcal{F}_1, \mathcal{F}_2 \subseteq \binom{[n]}{k}$ , the pair  $(\mathcal{F}_1, \mathcal{F}_2)$  is called cross-intersecting if for every two sets  $F_1 \in \mathcal{F}_1$  and  $F_2 \in \mathcal{F}_2$ , it holds that  $F_1 \cap F_2 \neq \emptyset$ . For a real  $\varepsilon \in [0, 1]$ , we say that the pair  $(\mathcal{F}_1, \mathcal{F}_2)$  is  $\varepsilon$ -close to cross-intersecting if it is possible to make  $(\mathcal{F}_1, \mathcal{F}_2)$  cross-intersecting by removing at most  $\varepsilon \cdot \binom{n}{k}$  of the sets of  $\mathcal{F}_1$  and  $\mathcal{F}_2$ . Otherwise, we say that  $(\mathcal{F}_1, \mathcal{F}_2)$  is  $\varepsilon$ -far from cross-intersecting. Note that if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting, then the pair  $(\mathcal{F}, \mathcal{F})$  is  $\varepsilon$ -far from cross-intersecting.

A function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  is called intersecting,  $\varepsilon$ -close to intersecting, and  $\varepsilon$ -far from intersecting, if the family  $f^{-1}(1)$  of the sets of  $\binom{[n]}{k}$  that are mapped by  $f$  to 1 is, respectively, intersecting,  $\varepsilon$ -close to intersecting, and  $\varepsilon$ -far from intersecting. For reals  $\varepsilon_1, \varepsilon_2 \in [0, 1]$  with  $\varepsilon_1 \leq \varepsilon_2$ , let  $\text{INTERSECTING}_{n,k,\varepsilon_1,\varepsilon_2}$  denote the tolerant property testing problem that given query access to a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$ , asks to distinguish between the case that  $f$  is  $\varepsilon_1$ -close to intersecting and the case that  $f$  is  $\varepsilon_2$ -far from intersecting. We will be particularly interested in the case of  $\varepsilon_1 = 0$ , hence for a real  $\varepsilon \in [0, 1]$ , we let  $\text{INTERSECTING}_{n,k,\varepsilon}$  denote the  $\text{INTERSECTING}_{n,k,0,\varepsilon}$  problem.

## 2.2 Chernoff–Hoeffding Bound

We will need the following version of the Chernoff–Hoeffding bound.

► **Theorem 5.** *For an integer  $m$  and a real  $p \in (0, 1)$ , let  $X_1, \dots, X_m$  be independent binary random variables satisfying  $\Pr[X_i = 1] = p$  and  $\Pr[X_i = 0] = 1 - p$  for all  $i \in [m]$ , and put  $\bar{X} = \frac{1}{m} \cdot \sum_{i=1}^m X_i$ . Then, for any  $\mu \geq 0$ ,*

1. *if  $p \leq \mu$ , then  $\Pr[\bar{X} \geq 2\mu] \leq e^{-m \cdot \mu/3}$ , and*
2. *if  $p \geq \mu$ , then  $\Pr[\bar{X} \leq \mu/2] \leq e^{-m \cdot \mu/8}$ .*

Note that the assertion of Theorem 5 for  $p = \mu$  follows from standard statements of the Chernoff–Hoeffding bound (see, e.g., [11, Theorem 2.3]). The cases of  $p < \mu$  and  $p > \mu$  stem by monotonicity.

### 3 Two-Sided Error Tester

In this section, we present and analyze a tolerant non-adaptive two-sided error tester for the  $\text{INTERSECTING}_{n,k,\varepsilon_1,\varepsilon_2}$  problem and prove the following result. Its special case with  $\varepsilon_1 = 0$  yields Theorem 1.

► **Theorem 6.** *For every integer  $r \geq 2$ , there exist constants  $c_1 = c_1(r)$  and  $c_2 = c_2(r)$  for which the following holds. For all sufficiently large integers  $n$  and  $k$  with  $n \geq 2k$  and for all reals  $\varepsilon_1, \varepsilon_2 \in [0, 1)$  with  $\varepsilon_2 \geq 4 \cdot \varepsilon_1 + c_1 \cdot (\frac{k}{n})^r$ , there exists a tolerant non-adaptive two-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon_1,\varepsilon_2}$  with at most  $c_2 \cdot \frac{\ln n}{\varepsilon_2}$  queries, running time polynomial in  $n$ , and success probability at least  $2/3$ .*

A crucial ingredient in the proof of Theorem 6 is the following theorem due to Dinur and Friedgut [3]. Here, a family  $\mathcal{J}$  of subsets of  $[n]$  is called a  $j$ -junta over  $[n]$  if there exists a set  $J \subseteq [n]$  of size  $|J| = j$  such that the membership of a set  $F$  in  $\mathcal{J}$  depends only on  $F \cap J$ .

► **Theorem 7** ([3]). *For every integer  $r \geq 2$ , there exist constants  $j = j(r)$  and  $a = a(r)$  for which the following holds. For all integers  $n$  and  $k$  with  $n > 2k$  and  $k > j$ , if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is intersecting, then there exists an intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$  such that  $|\mathcal{F} \setminus \mathcal{J}| \leq a \cdot \binom{n-r}{k-r}$ .*

We are ready to prove Theorem 6.

**Proof of Theorem 6.** Fix an integer  $r \geq 2$ , and let  $j = j(r)$  and  $a = a(r)$  be the constants given in Theorem 7. Let  $n$  and  $k$  be sufficiently large integers. Since the theorem trivially holds for  $n = 2k$  (with an appropriate  $c_1$ ), it may be assumed that  $n > 2k$  and  $k > j$ . For an integer  $m$  and for reals  $\varepsilon_1, \varepsilon_2 \in [0, 1)$  with  $\varepsilon_1 \leq \varepsilon_2$ , consider the following tester for the  $\text{INTERSECTING}_{n,k,\varepsilon_1,\varepsilon_2}$  problem.

Input: Query access to a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$ .

1. Pick  $m$  sets  $G_1, \dots, G_m$  uniformly and independently at random from  $\binom{[n]}{k}$ .
2. Query  $f$  for the value of  $f(G_i)$  for each  $i \in [m]$ .
3. For each intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , let  $\alpha_{\mathcal{J}}$  denote the fraction of the chosen sets  $G_i$  that are mapped by  $f$  to 1 and do not lie in  $\mathcal{J}$ , that is,

$$\alpha_{\mathcal{J}} = \frac{1}{m} \cdot |\{i \in [m] \mid f(G_i) = 1 \text{ and } G_i \notin \mathcal{J}\}|.$$

4. If there exists an intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$  with  $\alpha_{\mathcal{J}} \leq \frac{\varepsilon_2}{2}$ , then accept, and otherwise reject.

## 35:8 Testing Intersectingness of Uniform Families

The above tester is clearly non-adaptive. For the analysis of its success probability, suppose that

$$\varepsilon_2 \geq 4 \cdot \left( \varepsilon_1 + a \cdot \left( \frac{k}{n} \right)^r \right) \quad \text{and} \quad m \geq \frac{12 \cdot (j \cdot \ln n + 2^j + 2)}{\varepsilon_2}. \quad (1)$$

Let  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  be an input function, and consider the family  $\mathcal{F} = f^{-1}(1)$ . Our goal is to prove that if  $\mathcal{F}$  is  $\varepsilon_1$ -close to intersecting then the tester accepts  $f$  with probability at least  $2/3$ , and that if  $\mathcal{F}$  is  $\varepsilon_2$ -far from intersecting then the tester rejects  $f$  with probability at least  $2/3$ .

For each intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , let  $p_{\mathcal{J}}$  denote the fraction of the sets of  $\binom{[n]}{k}$  that are mapped by  $f$  to 1 and do not lie in  $\mathcal{J}$ , that is,

$$p_{\mathcal{J}} = \frac{1}{\binom{[n]}{k}} \cdot \left| \left\{ G \in \binom{[n]}{k} \mid f(G) = 1 \text{ and } G \notin \mathcal{J} \right\} \right|.$$

We shall prove that, with high probability, the values of  $p_{\mathcal{J}}$  are well approximated by the quantities  $\alpha_{\mathcal{J}}$  that our tester computes. Let  $\mathcal{E}$  denote the event that for every intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , it holds that

1. if  $p_{\mathcal{J}} \leq \frac{\varepsilon_2}{4}$ , then  $\alpha_{\mathcal{J}} < \frac{\varepsilon_2}{2}$ , and
2. if  $p_{\mathcal{J}} \geq \varepsilon_2$ , then  $\alpha_{\mathcal{J}} > \frac{\varepsilon_2}{2}$ .

For a fixed intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , apply Item 1 of Theorem 5 with  $\mu = \frac{\varepsilon_2}{4}$  to obtain that if  $p_{\mathcal{J}} \leq \frac{\varepsilon_2}{4}$ , then  $\alpha_{\mathcal{J}} \geq \frac{\varepsilon_2}{2}$  with probability at most  $e^{-m \cdot \varepsilon_2 / 12}$ . Further, apply Item 2 of Theorem 5 with  $\mu = \varepsilon_2$  to obtain that if  $p_{\mathcal{J}} \geq \varepsilon_2$ , then  $\alpha_{\mathcal{J}} \leq \frac{\varepsilon_2}{2}$  with probability at most  $e^{-m \cdot \varepsilon_2 / 8} \leq e^{-m \cdot \varepsilon_2 / 12}$ . The number of intersecting  $j$ -juntas over  $[n]$  is clearly bounded by  $\binom{[n]}{j} \cdot 2^{2^j}$ . Therefore, by the union bound, the probability that there exists an intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$  for which either  $p_{\mathcal{J}} \leq \frac{\varepsilon_2}{4}$  and  $\alpha_{\mathcal{J}} \geq \frac{\varepsilon_2}{2}$ , or  $p_{\mathcal{J}} \geq \varepsilon_2$  and  $\alpha_{\mathcal{J}} \leq \frac{\varepsilon_2}{2}$  does not exceed

$$\binom{[n]}{j} \cdot 2^{2^j} \cdot e^{-m \cdot \varepsilon_2 / 12} \leq n^j \cdot 2^{2^j} \cdot e^{-m \cdot \varepsilon_2 / 12} \leq e^{-2} < \frac{1}{3},$$

where the second inequality holds by our assumption on  $m$  given in (1). Therefore, the event  $\mathcal{E}$  occurs with probability at least  $2/3$ .

It suffices to show now that if the event  $\mathcal{E}$  occurs, then the answer of our tester is correct. Suppose first that  $\mathcal{F}$  is  $\varepsilon_1$ -close to intersecting. By definition, there exists an intersecting family  $\mathcal{F}' \subseteq \binom{[n]}{k}$  such that  $|\mathcal{F} \setminus \mathcal{F}'| \leq \varepsilon_1 \cdot \binom{[n]}{k}$ . By Theorem 7, using  $n > 2k$  and  $k > j$ , there exists an intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$  such that  $|\mathcal{F}' \setminus \mathcal{J}| \leq a \cdot \binom{[n-r]}{k-r} \leq a \cdot \left( \frac{k}{n} \right)^r \cdot \binom{[n]}{k}$ . It thus follows that this  $\mathcal{J}$  satisfies

$$|\mathcal{F} \setminus \mathcal{J}| \leq \varepsilon_1 \cdot \binom{[n]}{k} + |\mathcal{F}' \setminus \mathcal{J}| \leq \varepsilon_1 \cdot \binom{[n]}{k} + a \cdot \left( \frac{k}{n} \right)^r \cdot \binom{[n]}{k} \leq \frac{\varepsilon_2}{4} \cdot \binom{[n]}{k},$$

where the last inequality relies on our assumption on  $\varepsilon_1$  and  $\varepsilon_2$  in (1). This implies that  $p_{\mathcal{J}} \leq \frac{\varepsilon_2}{4}$ , and since the event  $\mathcal{E}$  occurs, it follows that  $\alpha_{\mathcal{J}} < \frac{\varepsilon_2}{2}$ , hence our tester accepts  $f$ . Next, suppose that  $\mathcal{F}$  is  $\varepsilon_2$ -far from intersecting. This implies, for each intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , that  $p_{\mathcal{J}} > \varepsilon_2$ , as otherwise, one could remove at most  $\varepsilon_2 \cdot \binom{[n]}{k}$  of the sets of  $\mathcal{F}$  to obtain a sub-family of the intersecting family  $\mathcal{J}$ . Since the event  $\mathcal{E}$  occurs, each such  $\mathcal{J}$  satisfies  $\alpha_{\mathcal{J}} > \frac{\varepsilon_2}{2}$ , hence our tester rejects  $f$ .

Finally, let  $m$  be the smallest integer satisfying the condition in (1). Observe that the running time of our tester is polynomial in  $m$  and in the number of intersecting  $j$ -juntas over  $[n]$ , where  $j$  depends only on  $r$ . It follows that the running time is polynomial in  $n$  and  $1/\varepsilon_2$ , so by our assumption on  $\varepsilon_2$  in (1), it is polynomial in  $n$ . This completes the proof.  $\blacktriangleleft$



► **Remark 8.** The assumption on  $\varepsilon_1$  and  $\varepsilon_2$  in Theorem 6 can be weakened, by an almost identical proof, to  $\varepsilon_2 \geq (1 + \delta) \cdot \varepsilon_1 + c_1 \cdot \left(\frac{k}{n}\right)^r$  with an arbitrary  $\delta > 0$ . For simplicity of presentation, we omit the details.

We conclude this section with the observation that for all integers  $n$  and  $k$  with  $n \geq 2k$  and for all reals  $\varepsilon_1, \varepsilon_2 \in [0, 1]$  with  $\varepsilon_2 \geq 4 \cdot \left(\varepsilon_1 + \frac{k}{n}\right)$ , there exists an efficient tolerant non-adaptive two-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon_1,\varepsilon_2}$  with  $O\left(\frac{1}{\varepsilon_2}\right)$  queries and high success probability. To see this, consider the tester that given access to a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  picks  $O\left(\frac{1}{\varepsilon_2}\right)$  sets uniformly and independently at random from  $\binom{[n]}{k}$ , queries  $f$  on them, and computes the fraction  $\alpha$  of the chosen sets that are mapped by  $f$  to 1. If  $\alpha \leq \frac{\varepsilon_2}{2}$ , then the tester accepts, and otherwise it rejects.

For correctness, let  $p$  denote the fraction of the sets of  $\binom{[n]}{k}$  that are mapped by  $f$  to 1. If  $f$  is  $\varepsilon_1$ -close to intersecting, then  $f$  can be made intersecting by flipping at most  $\varepsilon_1 \cdot \binom{[n]}{k}$  of its values. By the Erdős–Ko–Rado theorem [4], the fraction of the sets of  $\binom{[n]}{k}$  in an intersecting family does not exceed  $\frac{k}{n}$ , hence  $p \leq \varepsilon_1 + \frac{k}{n} \leq \frac{\varepsilon_2}{4}$ . In this case, Theorem 5 yields that with high probability, it holds that  $\alpha \leq \frac{\varepsilon_2}{2}$  and the tester accepts. On the other hand, if  $f$  is  $\varepsilon_2$ -far from intersecting, then it holds that  $p > \varepsilon_2$ , so using again Theorem 5, it follows that with high probability, we have  $\alpha > \frac{\varepsilon_2}{2}$  and the tester rejects.

## 4 One-Sided Error Tester

In this section, we present non-adaptive one-sided error testers for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem. We start with the canonical tester, which is used to prove Theorems 2 and 3. We then offer another non-adaptive one-sided error tester, appropriate for integers  $n$  and  $k$  with  $n = \Theta(k)$ .

### 4.1 Canonical Tester

Consider the following tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem.

**CANONICAL TESTER** ( $n, k, m$ )

Input: Query access to a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$ .

1. Pick  $m$  sets  $G_1, \dots, G_m$  uniformly and independently at random from  $\binom{[n]}{k}$ .
2. Query  $f$  for the value of  $f(G_i)$  for each  $i \in [m]$ .
3. If for every two indices  $i_1, i_2 \in [m]$  with  $G_{i_1} \cap G_{i_2} = \emptyset$ , it holds that either  $f(G_{i_1}) = 0$  or  $f(G_{i_2}) = 0$ , then accept, and otherwise reject.

The description of **CANONICAL TESTER** immediately implies that it is non-adaptive and that it accepts every intersecting function with probability 1 and is thus one-sided error. In what follows, we analyze the number of queries  $m$  needed to reject with high probability any function  $\varepsilon$ -far from intersecting for  $\varepsilon \geq \Omega\left(\left(\frac{k^2}{n}\right)^r\right)$ , where  $r$  is a fixed integer. We start with the case of  $r = 2$ , for which we offer a simple analysis that yields a bound of  $O\left(\frac{1}{\varepsilon}\right)$  on the query complexity. We then consider the case of a general  $r$ , for which we obtain a bound of  $O\left(\frac{\ln k}{\varepsilon}\right)$  on the query complexity.

#### 4.1.1 The case $r = 2$

We prove the following result, which confirms Theorem 3.

## 35:10 Testing Intersectingness of Uniform Families

► **Theorem 9.** For all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \in [0, 1)$  with  $\varepsilon \geq 2 \cdot \left(\frac{k^2}{n}\right)^2$ , there exists a non-adaptive one-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  with  $O\left(\frac{1}{\varepsilon}\right)$  queries, running time polynomial in  $n$ , and success probability at least  $2/3$ .

The proof of Theorem 9 requires two simple lemmas. The first one, given below, may be viewed as an analogue of a lemma of [2] for uniform families. Here, for a collection  $\mathcal{M}$  of pairs of sets, we say that the pairs of  $\mathcal{M}$  are pairwise disjoint if no set lies in more than one pair of  $\mathcal{M}$ .

► **Lemma 10.** For integers  $n$  and  $k$  with  $n \geq 2k$  and for a real  $\varepsilon \in [0, 1)$ , if a family  $\mathcal{F} \subseteq \binom{[n]}{k}$  is  $\varepsilon$ -far from intersecting, then there exists a collection of more than  $\frac{\varepsilon}{2} \cdot \binom{n}{k}$  pairwise disjoint pairs  $(C, D)$  of sets of  $\mathcal{F}$  satisfying  $C \cap D = \emptyset$ .

**Proof.** Let  $\mathcal{F} \subseteq \binom{[n]}{k}$  be a family  $\varepsilon$ -far from intersecting, and consider a maximal collection  $\mathcal{M}$  (with respect to inclusion) of pairwise disjoint pairs  $(C, D)$  of sets of  $\mathcal{F}$  satisfying  $C \cap D = \emptyset$ . The maximality of  $\mathcal{M}$  implies that  $\mathcal{F}$  can be made intersecting by removing all the  $2 \cdot |\mathcal{M}|$  sets that lie in the pairs of  $\mathcal{M}$ . Since  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting, it follows that  $2 \cdot |\mathcal{M}| > \varepsilon \cdot \binom{n}{k}$ . This completes the proof. ◀

Using Lemma 10, we obtain the following.

► **Lemma 11.** For integers  $n$  and  $k$  with  $n \geq 2k$  and for a real  $\varepsilon \in [0, 1)$ , let  $\mathcal{F} \subseteq \binom{[n]}{k}$  be a family  $\varepsilon$ -far from intersecting with  $|\mathcal{F}| > k^2 \cdot \binom{n-2}{k-2}$ . Then, more than  $\frac{\varepsilon}{2} \cdot \binom{n}{k}$  of the sets of  $\mathcal{F}$  are disjoint from at least  $\frac{1}{2} \cdot (|\mathcal{F}| - k^2 \cdot \binom{n-2}{k-2})$  of the sets of  $\mathcal{F}$ .

**Proof.** Let  $\mathcal{F} \subseteq \binom{[n]}{k}$  be a family  $\varepsilon$ -far from intersecting. By Lemma 10, there exists a collection  $\mathcal{M}$  of more than  $\frac{\varepsilon}{2} \cdot \binom{n}{k}$  pairwise disjoint pairs  $(C, D)$  of sets of  $\mathcal{F}$  satisfying  $C \cap D = \emptyset$ . Fix a pair  $(C, D) \in \mathcal{M}$ . The disjointness of  $C$  and  $D$  implies that every set in  $\binom{[n]}{k}$  that intersects both  $C$  and  $D$  includes some element of  $C$  and a different element of  $D$ . It follows that the number of sets in  $\binom{[n]}{k}$  that intersect both  $C$  and  $D$  does not exceed  $k^2 \cdot \binom{n-2}{k-2}$ . Therefore, either  $C$  or  $D$  is disjoint from at least  $\frac{1}{2} \cdot (|\mathcal{F}| - k^2 \cdot \binom{n-2}{k-2})$  of the sets of  $\mathcal{F}$ . Since the pairs of  $\mathcal{M}$  are pairwise disjoint, it follows that more than  $\frac{\varepsilon}{2} \cdot \binom{n}{k}$  sets of  $\mathcal{F}$  are disjoint from at least  $\frac{1}{2} \cdot (|\mathcal{F}| - k^2 \cdot \binom{n-2}{k-2})$  of the sets of  $\mathcal{F}$ , as required. ◀

We are ready to prove Theorem 9.

**Proof of Theorem 9.** For integers  $n, k, m$  and for a real  $\varepsilon \in [0, 1)$ , consider the  $\text{CANONICAL TESTER}(n, k, m)$  for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem. As previously mentioned, this tester is non-adaptive and one-sided error. To establish its correctness, it suffices to prove that if a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  is  $\varepsilon$ -far from intersecting, then the  $m$  sets picked by the tester include with probability at least  $2/3$  two disjoint sets that lie in  $f^{-1}(1)$ . Indeed, this implies that the tester rejects such an  $f$  with the desired probability.

For a real  $\varepsilon \geq 2 \cdot \left(\frac{k^2}{n}\right)^2$ , let  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  be a function  $\varepsilon$ -far from intersecting. Consider the family  $\mathcal{F} = f^{-1}(1)$ , and note that  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting. In particular, it holds that  $|\mathcal{F}| > \varepsilon \cdot \binom{n}{k}$ , because  $\mathcal{F}$  can be made intersecting by removing all of its sets. This implies that

$$|\mathcal{F}| - k^2 \cdot \binom{n-2}{k-2} > \varepsilon \cdot \binom{n}{k} - k^2 \cdot \left(\frac{k}{n}\right)^2 \cdot \binom{n}{k} \geq \frac{\varepsilon}{2} \cdot \binom{n}{k}, \quad (2)$$

where the second inequality relies on our assumption on  $\varepsilon$ . We say that a set of  $\binom{[n]}{k}$  is useful if it lies in  $\mathcal{F}$  and is disjoint from at least  $\frac{\varepsilon}{4} \cdot \binom{n}{k}$  of the sets of  $\mathcal{F}$ . By Lemma 11, combined with (2), the number of useful sets is greater than  $\frac{\varepsilon}{2} \cdot \binom{n}{k}$ . Hence, a random set picked uniformly from  $\binom{[n]}{k}$  is useful with probability at least  $\varepsilon/2$ .

Now, consider the  $m$  sets chosen by CANONICAL TESTER. The probability that no useful set is picked throughout the first  $4/\varepsilon$  choices does not exceed  $(1 - \varepsilon/2)^{4/\varepsilon} \leq e^{-2} < 1/6$ . Therefore, with probability at least  $5/6$ , these choices include a useful set  $A$ . Since  $A$  is useful, it is disjoint from at least  $\frac{\varepsilon}{4} \cdot \binom{n}{k}$  of the sets of  $\mathcal{F}$ . Therefore, once such a set  $A$  is chosen, the probability that a random set picked uniformly from  $\binom{[n]}{k}$  lies in  $\mathcal{F}$  and is disjoint from  $A$  is at least  $\varepsilon/4$ . It follows that the probability that no such set is picked throughout the next  $8/\varepsilon$  choices does not exceed  $(1 - \varepsilon/4)^{8/\varepsilon} \leq e^{-2} < 1/6$ . By the union bound, for  $m = 12/\varepsilon$ , the  $m$  sets picked by our tester include with probability at least  $2/3$  two disjoint sets that lie in  $\mathcal{F}$ . Observe that for this choice of  $m$ , the running time of our tester is polynomial in  $n$  and  $1/\varepsilon$ . By our assumption on  $\varepsilon$ , the running time is polynomial in  $n$ , as desired. ◀

### 4.1.2 General $r$

We proceed by analyzing the more nuanced case in which  $\varepsilon \geq \Omega((k^2/n)^r)$  for a general integer  $r$ . We prove the following result, which confirms Theorem 2.

► **Theorem 12.** *For every integer  $r \geq 1$ , there exist constants  $c_1 = c_1(r)$  and  $c_2 = c_2(r)$  for which the following holds. For all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \in [0, 1]$  with  $\varepsilon \geq c_1 \cdot (\frac{k^2}{n})^r$ , there exists a non-adaptive one-sided error tester for  $\text{INTERSECTING}_{n,k,\varepsilon}$  with at most  $c_2 \cdot \frac{\ln k}{\varepsilon}$  queries, running time polynomial in  $n$ , and success probability at least  $2/3$ .*

The proof of Theorem 12 requires a few lemmas, which involve the following definition.

► **Definition 13.** *For integers  $n$  and  $k$  with  $n \geq 2k$ , a family  $\mathcal{F} \subseteq \binom{[n]}{k}$ , a set  $A \subseteq [n]$ , and a set  $B \subseteq A$ , we let  $\mathcal{F}(A \downarrow B)$  denote the family of sets of  $\mathcal{F}$  whose intersection with  $A$  is  $B$ , that is,*

$$\mathcal{F}(A \downarrow B) = \{F \in \mathcal{F} \mid F \cap A = B\}.$$

We say that the set  $A$   $\varepsilon$ -captures  $\mathcal{F}$  if the number of sets of  $\mathcal{F}$  that are disjoint from  $A$  is smaller than  $\varepsilon \cdot \binom{n}{k}$ , equivalently,  $|\mathcal{F}(A \downarrow \emptyset)| < \varepsilon \cdot \binom{n}{k}$ .

The following lemma shows that if two families are far from cross-intersecting, then for every small set  $A$ , it is possible to restrict the families to disjoint intersections with  $A$ , so that the obtained restrictions are still far from cross-intersecting. The proof can be found in the full version of the paper.

► **Lemma 14.** *For an integer  $r \geq 0$ , integers  $n$  and  $k$  with  $n \geq 2k$ , and a real  $\varepsilon \in [0, 1]$ , let  $\mathcal{F}_1, \mathcal{F}_2 \subseteq \binom{[n]}{k}$  be two families such that the pair  $(\mathcal{F}_1, \mathcal{F}_2)$  is  $\varepsilon$ -far from cross-intersecting, and let  $A \subseteq [n]$  be a set of size  $|A| \leq r$ . Then, there exist two sets  $B, C \subseteq A$  with  $B \cap C = \emptyset$  for which the pair  $(\mathcal{F}_1(A \downarrow B), \mathcal{F}_2(A \downarrow C))$  is  $\frac{\varepsilon}{3^{r-t}}$ -far from cross-intersecting. Moreover, if  $A$   $\frac{\varepsilon}{3^r}$ -captures  $\mathcal{F}_2$ , then the guaranteed set  $C$  is not empty.*

As a consequence of Lemma 14, we obtain the following.

► **Lemma 15.** *For integers  $r, t \geq 0$ , integers  $n$  and  $k$  with  $n \geq 2k$ , and a real  $\varepsilon \in [0, 1]$ , let  $\mathcal{F}_1, \mathcal{F}_2 \subseteq \binom{[n]}{k}$  be families such that the pair  $(\mathcal{F}_1, \mathcal{F}_2)$  is  $\varepsilon$ -far from cross-intersecting. Then, there exist sets  $A \subseteq [n]$  and  $B, C \subseteq A$  with  $B \cap C = \emptyset$ , such that*

1. *the pair  $(\mathcal{F}_1(A \downarrow B), \mathcal{F}_2(A \downarrow C))$  is  $\frac{\varepsilon}{3^{r-t}}$ -far from cross-intersecting, and*
  2. *either there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r-t}}$ -captures  $\mathcal{F}_2(A \downarrow C)$ , or  $|C| \geq t$ .*
- Moreover, if  $\varepsilon \geq 3^{r-t} \cdot (\frac{k}{n})^t$ , then the guaranteed set  $C$  satisfies  $|C| < t$ .<sup>1</sup>

<sup>1</sup> Note that by combining the second item of the lemma with the “moreover” part, it follows that if  $\varepsilon \geq 3^{r-t} \cdot (\frac{k}{n})^t$ , then there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r-t}}$ -captures  $\mathcal{F}_2(A \downarrow C)$ .

## 35:12 Testing Intersectingness of Uniform Families

**Proof.** Fix an integer  $r \geq 0$ . We start by proving the first part of the lemma, i.e., the existence of sets  $A \subseteq [n]$  and  $B, C \subseteq A$  with  $B \cap C = \emptyset$  satisfying Items 1 and 2. To do so, we apply induction on  $t$ . The result clearly holds for  $t = 0$ , as follows from the choice  $A = B = C = \emptyset$  (for which we have  $\mathcal{F}_1 = \mathcal{F}_1(A \downarrow_B)$ ,  $\mathcal{F}_2 = \mathcal{F}_2(A \downarrow_C)$ , and  $|C| \geq 0$ ).

Now, take  $t \geq 1$ , and assume that the result holds for  $t - 1$ . To prove it for  $t$ , let  $\mathcal{F}_1, \mathcal{F}_2 \subseteq \binom{[n]}{k}$  be families such that the pair  $(\mathcal{F}_1, \mathcal{F}_2)$  is  $\varepsilon$ -far from cross-intersecting. By the induction hypothesis, there exist sets  $A \subseteq [n]$  and  $B, C \subseteq A$  with  $B \cap C = \emptyset$ , such that the families  $\mathcal{H}_1 = \mathcal{F}_1(A \downarrow_B)$  and  $\mathcal{H}_2 = \mathcal{F}_2(A \downarrow_C)$  satisfy that

1. the pair  $(\mathcal{H}_1, \mathcal{H}_2)$  is  $\frac{\varepsilon}{3^{r \cdot (t-1)}}$ -far from cross-intersecting, and
  2. either there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r \cdot (t-1)}}$ -captures  $\mathcal{H}_2$ , or  $|C| \geq t - 1$ .
- By  $\frac{\varepsilon}{3^{r \cdot t}} \leq \frac{\varepsilon}{3^{r \cdot (t-1)}}$ , it follows from Item 1 that the pair  $(\mathcal{H}_1, \mathcal{H}_2)$  is  $\frac{\varepsilon}{3^{r \cdot t}}$ -far from cross-intersecting. Therefore, if there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r \cdot t}}$ -captures  $\mathcal{H}_2$ , then the result holds for  $t$  with the same sets  $A, B, C$ , and we are done. Otherwise, there exists a set  $A' \subseteq [n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r \cdot t}}$ -captures  $\mathcal{H}_2$ . In particular, there exists a subset of  $[n] \setminus A$  of size at most  $r$  that  $\frac{\varepsilon}{3^{r \cdot (t-1)}}$ -captures  $\mathcal{H}_2$ , hence by Item 2 above, it follows that  $|C| \geq t - 1$ .

We apply Lemma 14 with the pair  $(\mathcal{H}_1, \mathcal{H}_2)$ , which is  $\frac{\varepsilon}{3^{r \cdot (t-1)}}$ -far from cross-intersecting, and with the set  $A'$ . Letting  $B', C' \subseteq A'$  be the sets with  $B' \cap C' = \emptyset$  guaranteed by the lemma, it follows that the pair  $(\mathcal{H}_1(A' \downarrow_{B'}), \mathcal{H}_2(A' \downarrow_{C'}))$  is  $\tilde{\varepsilon}$ -far from cross-intersecting for  $\tilde{\varepsilon} = \frac{1}{3^r} \cdot \frac{1}{3^{r \cdot (t-1)}} = \frac{1}{3^{r \cdot t}}$ . Moreover, since  $A'$   $\tilde{\varepsilon}$ -captures  $\mathcal{H}_2$ , it follows from the lemma that  $C'$  is not empty.

To complete the argument, define  $A'' = A \cup A'$ ,  $B'' = B \cup B'$ , and  $C'' = C \cup C'$ . Recalling that  $A \cap A' = \emptyset$ ,  $B \cap C = \emptyset$ , and  $B' \cap C' = \emptyset$ , we obtain that  $B'' \cap C'' = \emptyset$  and that

$$|C''| = |C| + |C'| \geq (t - 1) + 1 = t.$$

We further have  $\mathcal{H}_1(A' \downarrow_{B'}) = \mathcal{F}_1(A' \downarrow_{B''})$  and  $\mathcal{H}_2(A' \downarrow_{C'}) = \mathcal{F}_2(A' \downarrow_{C''})$ . Hence, the sets  $A'', B'', C''$  satisfy the required properties with respect to  $\mathcal{F}_1, \mathcal{F}_2$  and  $t$ . This completes the proof of the first part of the lemma.

We finally show that if  $\varepsilon \geq 3^{r \cdot t} \cdot \left(\frac{k}{n}\right)^t$ , then the guaranteed set  $C$  satisfies  $|C| < t$ . To establish the contrapositive statement, let  $A, B, C$  be the sets guaranteed by the lemma for two families  $\mathcal{F}_1, \mathcal{F}_2 \subseteq \binom{[n]}{k}$ , and suppose that  $|C| \geq t$ . Since the pair  $(\mathcal{F}_1(A \downarrow_B), \mathcal{F}_2(A \downarrow_C))$  is  $\frac{\varepsilon}{3^{r \cdot t}}$ -far from cross-intersecting, it follows that  $|\mathcal{F}_2(A \downarrow_C)| > \frac{\varepsilon}{3^{r \cdot t}} \cdot \binom{n}{k}$ . On the other hand, each set of  $\mathcal{F}_2(A \downarrow_C)$  contains the set  $C$ , hence  $|\mathcal{F}_2(A \downarrow_C)| \leq \binom{n-t}{k-t} \leq \left(\frac{k}{n}\right)^t \cdot \binom{n}{k}$ . By combining the two inequalities, we obtain that  $\varepsilon < 3^{r \cdot t} \cdot \left(\frac{k}{n}\right)^t$ . This completes the proof.  $\blacktriangleleft$

The following lemma constitutes a key ingredient in our analysis of the canonical tester. Its proof can be found in the full version of the paper.

**► Lemma 16.** *For every integer  $r \geq 1$ , there exists a constant  $c = c(r)$ , such that for all integers  $n$  and  $k$  with  $n \geq 2k$  and for any real  $\varepsilon \in [0, 1)$  with  $\varepsilon \geq 2 \cdot \left(\frac{k^2}{n}\right)^r$ , the following holds. Let  $\mathcal{F} \subseteq \binom{[n]}{k}$  be a family, and suppose that there exist sets  $A \subseteq [n]$  and  $B, C \subseteq A$  with  $B \cap C = \emptyset$ , such that the pair  $(\mathcal{F}(A \downarrow_B), \mathcal{F}(A \downarrow_C))$  is  $\varepsilon$ -far from cross-intersecting, and there is no subset of  $[n] \setminus A$  of size at most  $r - 1$  that  $\varepsilon$ -captures  $\mathcal{F}(A \downarrow_C)$ . Then, if at least  $c \cdot \frac{\ln k}{\varepsilon}$  sets are chosen uniformly and independently at random from  $\binom{[n]}{k}$ , then the probability that they include two disjoint sets that lie in  $\mathcal{F}$  is at least  $2/3$ .*

We are ready to prove Theorem 12.

**Proof of Theorem 12.** For integers  $n, k, m$  and for a real  $\varepsilon \in [0, 1]$ , consider the CANONICAL TESTER  $(n, k, m)$  for the INTERSECTING $_{n,k,\varepsilon}$  problem. As previously mentioned, this tester is non-adaptive and one-sided error. Fix an integer  $r \geq 1$ , let  $c = c(r)$  be the constant given in Lemma 16, and suppose that

$$\varepsilon \geq 2 \cdot 3^{r^2} \cdot \left(\frac{k^2}{n}\right)^r \quad \text{and} \quad m \geq c \cdot 3^{r^2} \cdot \frac{\ln k}{\varepsilon}. \quad (3)$$

It suffices to prove that if a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  is  $\varepsilon$ -far from intersecting, then the  $m$  sets picked by our tester include with probability at least  $2/3$  two disjoint sets that lie in  $f^{-1}(1)$ . Indeed, this implies that the tester rejects such an  $f$  with the desired probability.

Let  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  be a function  $\varepsilon$ -far from intersecting. Set  $\mathcal{F} = f^{-1}(1)$ , and note that the family  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting, hence the pair  $(\mathcal{F}, \mathcal{F})$  is  $\varepsilon$ -far from cross-intersecting. Apply Lemma 15 with  $\mathcal{F}_1 = \mathcal{F}_2 = \mathcal{F}$  and with  $t = r$ , to obtain that there exist sets  $A \subseteq [n]$  and  $B, C \subseteq A$  with  $B \cap C = \emptyset$ , such that for  $\tilde{\varepsilon} = \frac{\varepsilon}{3^{r^2}}$ , it holds that

1. the pair  $(\mathcal{F}(A \downarrow_B), \mathcal{F}(A \downarrow_C))$  is  $\tilde{\varepsilon}$ -far from cross-intersecting, and
  2. either there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\tilde{\varepsilon}$ -captures  $\mathcal{F}_2(A \downarrow_C)$ , or  $|C| \geq r$ .
- Moreover, our assumption on  $\varepsilon$  in (3) clearly implies that  $\varepsilon \geq 3^{r^2} \cdot \left(\frac{k}{n}\right)^r$ , hence it follows from the lemma that  $|C| < r$ . We thus derive from Item 2 above that there is no subset of  $[n] \setminus A$  of size at most  $r$  that  $\tilde{\varepsilon}$ -captures  $\mathcal{F}(A \downarrow_C)$ . This obviously implies that there is no subset of  $[n] \setminus A$  of size at most  $r - 1$  that  $\tilde{\varepsilon}$ -captures  $\mathcal{F}(A \downarrow_C)$ .

Now, by our assumptions in (3), it holds that  $\tilde{\varepsilon} \geq 2 \cdot \left(\frac{k^2}{n}\right)^r$  and  $m \geq c \cdot \frac{\ln k}{\tilde{\varepsilon}}$ . Therefore, we can apply Lemma 16 with the family  $\mathcal{F}$ , the sets  $A, B, C$ , the integer  $r$ , and the real  $\tilde{\varepsilon}$ , to obtain that with probability at least  $2/3$ , the  $m$  sets picked by our tester include two disjoint sets that lie in  $\mathcal{F}$ , as required.

Finally, let  $m$  be the smallest integer satisfying the condition in (3). Observe that the running time of our tester is polynomial in  $n$  and  $m$ , hence it is polynomial in  $n$  and  $1/\varepsilon$ . By our assumption on  $\varepsilon$  in (3), the running time is polynomial in  $n$ , and the proof is completed.  $\blacktriangleleft$

## 4.2 The case $n = \Theta(k)$

We turn our attention now to the INTERSECTING $_{n,k,\varepsilon}$  problem, where the integers  $n$  and  $k$  satisfy  $n = \alpha \cdot k$  for an arbitrary constant  $\alpha \geq 2$ . We first observe that for this range of parameters, the canonical tester is not effective, even for a constant  $\varepsilon$ . To see this, observe that two random sets, chosen uniformly and independently from  $\binom{[n]}{k}$ , are disjoint with probability  $\binom{n-k}{k} / \binom{n}{k}$ . For  $n = \alpha \cdot k$ , with  $\alpha \geq 2$  being a constant, this probability decreases exponentially in  $n$ . Therefore, a collection of random sets from  $\binom{[n]}{k}$  is unlikely to include even a single pair of disjoint sets, unless their number grows exponentially in  $n$ . As a result, the canonical tester does not provide a useful upper bound on the query complexity of the INTERSECTING $_{n,k,\varepsilon}$  problem in this setting.

To overcome this difficulty, we consider a slightly different tester for the INTERSECTING $_{n,k,\varepsilon}$  problem, which picks random pairs of disjoint sets from  $\binom{[n]}{k}$  and checks if at least one of the pairs demonstrates a violation of intersectingness for the tested function. This tester allows us to prove the following result.

**► Theorem 17.** *For all reals  $\alpha \geq 2$  and  $\varepsilon \in (0, 1)$ , there exists some  $c = c(\alpha, \varepsilon)$ , such that for all sufficiently large integers  $n$  and  $k$  with  $n = \alpha \cdot k$ , there exists a non-adaptive one-sided error tester for INTERSECTING $_{n,k,\varepsilon}$  with  $c$  queries and success probability at least  $2/3$ .*

Theorem 17 is obtained using the following result, that was proved (in a generalized form) by Friedgut and Regev [7].

► **Theorem 18** ([7]). *For all reals  $\alpha > 2$  and  $\varepsilon \in (0, 1)$ , there exist a real  $\delta = \delta(\alpha, \varepsilon) > 0$  and an integer  $j = j(\alpha, \varepsilon)$ , such that for all sufficiently large integers  $n$  and  $k$  with  $n = \alpha \cdot k$ , the following holds. Suppose that  $\mathcal{F} \subseteq \binom{[n]}{k}$  is a family, such that for every intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$ , it holds that  $|\mathcal{F} \setminus \mathcal{J}| > \varepsilon \cdot \binom{n}{k}$ . Then, a random unordered pair  $\{A, B\}$  of two disjoint sets of  $\binom{[n]}{k}$ , chosen uniformly from all such pairs, satisfies  $A \in \mathcal{F}$  and  $B \in \mathcal{F}$  with probability at least  $\delta$ .*

**Proof of Theorem 17.** Fix  $\alpha \geq 2$  and  $\varepsilon \in (0, 1)$ , and let  $n$  and  $k$  be sufficiently large integers with  $n = \alpha \cdot k$ . For an integer  $m$ , consider the following tester for the  $\text{INTERSECTING}_{n,k,\varepsilon}$  problem.

Input: Query access to a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$ .

1. Pick  $m$  unordered pairs  $\{A_1, B_1\}, \dots, \{A_m, B_m\}$  of sets of  $\binom{[n]}{k}$  with  $A_i \cap B_i = \emptyset$  for all  $i \in [m]$  uniformly and independently at random.
2. Query  $f$  for the values of  $f(A_i)$  and  $f(B_i)$  for each  $i \in [m]$ .
3. If for each  $i \in [m]$ , it holds that  $f(A_i) = 0$  or  $f(B_i) = 0$ , then accept, and otherwise reject.

The above tester is clearly non-adaptive. Since it accepts every intersecting function with probability 1, its error is one-sided. It suffices to show that there exists  $m = m(\alpha, \varepsilon)$ , such that if a function  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  is  $\varepsilon$ -far from intersecting, then our tester rejects it with probability at least  $2/3$ . Let  $f : \binom{[n]}{k} \rightarrow \{0, 1\}$  be a function  $\varepsilon$ -far from intersecting.

Consider first the simple case of  $\alpha = 2$ . Here, the fact that  $f$  is  $\varepsilon$ -far from intersecting implies that more than  $\varepsilon \cdot \binom{n}{k}$  of the  $\frac{1}{2} \cdot \binom{n}{k}$  unordered pairs of disjoint sets of  $\binom{[n]}{k}$  violate the intersectingness. For  $m = 1/\varepsilon$ , the probability that our tester accepts  $f$  is at most  $(1 - 2\varepsilon)^m = (1 - 2\varepsilon)^{1/\varepsilon} \leq e^{-2}$ . Hence, with the complement probability, which exceeds  $2/3$ , our tester rejects  $f$ , as desired.

Next, suppose that  $\alpha > 2$ , let  $\delta = \delta(\alpha, \varepsilon) > 0$  and  $j = j(\alpha, \varepsilon)$  be the constants given in Theorem 18, and put  $\mathcal{F} = f^{-1}(1)$ . Since  $\mathcal{F}$  is  $\varepsilon$ -far from intersecting, it follows that every intersecting  $j$ -junta  $\mathcal{J}$  over  $[n]$  satisfies  $|\mathcal{F} \setminus \mathcal{J}| > \varepsilon \cdot \binom{n}{k}$ . By Theorem 18, at least  $\delta$  fraction of the unordered pairs of disjoint sets of  $\binom{[n]}{k}$  violate the intersectingness. Letting  $m = 2/\delta$ , the probability that our tester accepts  $f$  is at most  $(1 - \delta)^m = (1 - \delta)^{2/\delta} \leq e^{-2}$ . As before, it follows that with probability at least  $2/3$ , our tester rejects  $f$ , and we are done. ◀

---

## References

- 1 Hou Tin Chau, David Ellis, Ehud Friedgut, and Noam Lifshitz. On the maximum degree of induced subgraphs of the Kneser graph. *arXiv*, abs/2312.06370, 2023. [arXiv:2312.06370](#).
- 2 Xi Chen, Anindya De, Yuhao Li, Shivam Nadimpalli, and Rocco A. Servedio. Testing intersecting and union-closed families. In *Proc. of the 15th Innovations in Theoretical Computer Science Conference (ITCS'24)*, pages 33:1–33:23, 2024.
- 3 Irit Dinur and Ehud Friedgut. Intersecting families are essentially contained in juntas. *Comb. Probab. Comput.*, 18(1–2):107–122, 2009.
- 4 Paul Erdős, Chao Ko, and Richard Rado. Intersection theorems for systems of finite sets. *Quart. J. Math.*, 12(1):313–320, 1961.
- 5 Eldar Fischer. A basic lower bound for property testing. *arXiv*, abs/2403.04999, 2024. [arXiv:2403.04999](#).
- 6 Peter Frankl and Andrey Kupavskii. Maximal degrees in subgraphs of Kneser graphs. *arXiv*, abs/2004.08718, 2020. [arXiv:2004.08718](#).

- 7 Ehud Friedgut and Oded Regev. Kneser graphs are like Swiss cheese. *Discrete Analysis*, 2:1–18, 2018.
- 8 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- 9 Martin Kneser. Aufgabe 360. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 58(2):27, 1955.
- 10 László Lovász. Kneser’s conjecture, chromatic number, and homotopy. *J. Comb. Theory, Ser. A*, 25(3):319–324, 1978.
- 11 Colin McDiarmid. Concentration. In *Probabilistic methods for algorithmic discrete mathematics*, volume 16 of *Algorithms Combin.*, pages 195–248. Springer, Berlin, 1998.
- 12 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006.
- 13 Roei Tell. A note on tolerant testing with one-sided error. In *Computational Complexity and Property Testing – On the Interplay Between Randomness and Computation*, pages 173–177. Springer, 2020.





# On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs

Lap Chi Lau ✉

Cheriton School of Computer Science, University of Waterloo, Canada

Dante Tjowasi ✉

Cheriton School of Computer Science, University of Waterloo, Canada

---

## Abstract

Houdré and Tetali defined a class of isoperimetric constants  $\varphi_p$  of graphs for  $0 \leq p \leq 1$ , and conjectured a Cheeger-type inequality for  $\varphi_{\frac{1}{2}}$  of the form

$$\lambda_2 \lesssim \varphi_{\frac{1}{2}} \lesssim \sqrt{\lambda_2},$$

where  $\lambda_2$  is the second smallest eigenvalue of the normalized Laplacian matrix. If true, the conjecture would be a strengthening of the hard direction of the classical Cheeger's inequality. Morris and Peres proved Houdré and Tetali's conjecture up to an additional log factor, using techniques from evolving sets. We present the following related results on this conjecture.

1. We provide a family of counterexamples to the conjecture of Houdré and Tetali, showing that the logarithmic factor is needed.
2. We match Morris and Peres's bound using standard spectral arguments.
3. We prove that Houdré and Tetali's conjecture is true for any constant  $p$  strictly bigger than  $\frac{1}{2}$ , which is also a strengthening of the hard direction of Cheeger's inequality.

Furthermore, our results can be extended to directed graphs using Chung's definition of eigenvalues for directed graphs.

**2012 ACM Subject Classification** Theory of computation → Random walks and Markov chains

**Keywords and phrases** Isoperimetric constant, Markov chains, Cheeger's inequality

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.36

**Category** RANDOM

**Funding** Supported by an NSERC Discovery Grant and a Math-funded Undergraduate Research Award from University of Waterloo.

**Acknowledgements** We thank Christian Houdré and Prasad Tetali for their encouragement and the anonymous reviewers for their helpful comments.

## 1 Introduction

Motivated by Talagrand's isoperimetric inequality for the hypercubes [20] (see Subsection 1.2), Houdré and Tetali [8] extended Talagrand's isoperimetric constant to general Markov chains and also to different exponents.

► **Definition 1** (Isoperimetric Constants for Markov Chains [8]). *Let  $(V, P, \pi)$  be an irreducible Markov chain with vertex set  $V$ , transition matrix  $P \in \mathbb{R}^{|V| \times |V|}$  and stationary distribution  $\pi : V \rightarrow \mathbb{R}_{\geq 0}$ . For any  $p \in (0, 1]$ , define the isoperimetric constant as*

$$\varphi_p(P) := \min_{S \subset V: \pi(S) \leq \frac{1}{2}} \varphi_p(S) := \min_{S \subset V: \pi(S) \leq \frac{1}{2}} \frac{\sum_{v \in S} \pi(v) \cdot \left( \sum_{u \in \bar{S}} P(v, u) \right)^p}{\pi(S)}.$$



© Lap Chi Lau and Dante Tjowasi;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 36; pp. 36:1–36:18



Leibniz International Proceedings in Informatics

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

## 36:2 On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs

Let  $\partial S := \{v \in S \mid \sum_{u \in \bar{S}} P(v, u) > 0\}$  be the inner vertex boundary of  $S$ . Then, for  $p = 0$ ,

$$\varphi_0(P) := \min_{S \subset V: \pi(S) \leq \frac{1}{2}} \varphi_0(S) := \min_{S \subset V: \pi(S) \leq \frac{1}{2}} \frac{\pi(\partial S)}{\pi(S)}.$$

Given an undirected graph or a directed graph  $G = (V, E)$ , let  $P_G$  be the transition matrix of the natural random walk, where  $P_G(v, u) = 1/\deg(v)$  in the undirected case and  $P_G(v, u) = 1/\deg^{\text{out}}(v)$  in the directed case. Then the isoperimetric constant  $\varphi_p(G)$  for the graph  $G$  is defined as  $\varphi_p(P_G)$ .

When  $p = 1$ , this is known as the Cheeger isoperimetric constant of a Markov chain (see e.g. [16]) or the edge conductance/expansion of a graph (see Section 2 for definition). When  $p = 0$ , this measures the vertex expansion of an undirected graph or a directed graph. Talagrand studied the case  $p = \frac{1}{2}$  and proved a lower bound on  $\varphi_{\frac{1}{2}}$  for Boolean hypercubes. One can view  $\varphi_{\frac{1}{2}}$  as a quantity that interpolates between edge conductance and vertex expansion, since it follows from the Cauchy-Schwarz inequality that  $\varphi_{\frac{1}{2}}(G)^2 \leq \varphi_1(G) \cdot \varphi_0(G)$ , and Talagrand used his lower bound to recover Margulis' theorem about edge conductance and vertex expansion on hypercubes (see Subsection 1.2).

For an undirected graph  $G$ , one can use the second smallest eigenvalue  $\lambda_2$  of the matrix  $I - P_G$  to give upper and lower bound on  $\varphi_1(G)$ . The classical Cheeger's inequality is

$$\lambda_2(I - P_G) \lesssim \varphi_1(G) \lesssim \sqrt{\lambda_2(I - P_G)}.$$

Houdré and Tetali conjectured that the same relations hold for  $\varphi_{\frac{1}{2}}(G)$  as well when the Markov chain is reversible.

► **Conjecture 2** ([8]). *Let  $(V, P, \pi)$  be an irreducible and reversible Markov chain. Then*

$$\lambda_2(I - P) \lesssim \varphi_{\frac{1}{2}}(P) \lesssim \sqrt{\lambda_2(I - P)}.$$

It is clear from the definition that  $\varphi_p(G)$  increases as  $p$  decreases, and thus  $\lambda_2 \lesssim \varphi_1(G) \leq \varphi_p(G)$  for all  $p < 1$ . Therefore, the Houdré-Tetali conjecture is a strengthening of the hard direction of Cheeger's inequality. It predicts that when the hard direction of Cheeger's inequality is tight for a graph  $G$  such that  $\varphi_1(G) \asymp \sqrt{\lambda_2}$ , then the graph must satisfy  $\varphi_1(G) \asymp \varphi_{\frac{1}{2}}(G)$ . Or, in other words, when  $\varphi_1(G) \ll \varphi_{\frac{1}{2}}(G)$  such as on the hypercube (see Remark 7) or on the dumbbell graphs, then the hard direction of Cheeger's inequality cannot be tight. So, the conjecture can be viewed as an improved Cheeger's inequality in the spirit of [11, 10] and this is a main motivation of this work.

Morris and Peres came close to proving the conjecture with an extra logarithmic factor.

► **Theorem 3** ([18]). *Let  $(V, P, \pi)$  be an irreducible and reversible Markov chain. Suppose that  $P(v, v) \geq \frac{1}{2}$  for all  $v \in V$ . Then*

$$\lambda_2(I - P) \gtrsim \frac{(\varphi_{\frac{1}{2}}(P))^2}{\log(1/\varphi_{\frac{1}{2}}(P))}.$$

Their proof is based on techniques from evolving sets. They lower bounded the "boundary gauge"  $\Psi$  using  $\varphi_{\frac{1}{2}}$ , and also upper bounded the mixing rate using  $\Psi$  so that they can relate  $\lambda_2$  and  $\varphi_{\frac{1}{2}}$ .

### 1.1 Our Results

We found counterexamples to the Houdré-Tetali conjecture, showing that the extra logarithmic factor is needed.

► **Theorem 4.** *There are irreducible and reversible Markov chains  $(V, P, \pi)$  with*

$$\lambda_2(I - P) \lesssim \frac{\log |V|}{|V|^2} \quad \text{and} \quad \varphi_{\frac{1}{2}}(P) \gtrsim \frac{\log |V|}{|V|} \quad \implies \quad \lambda_2(I - P) \lesssim \frac{(\varphi_{\frac{1}{2}}(P))^2}{\log(1/\varphi_{\frac{1}{2}}(P))}.$$

The counterexample is simple to describe, which is a weighted undirected graph with vertex set  $[n]$  and edge weight  $P(i, j)$  inversely proportional to  $\min\{|i - j|, n - |i - j|\}^3$ . See Section 4 for details.

On the positive side, we match the result of Morris and Peres using standard spectral arguments. We show that the simple sweep-cut algorithm can be used to output a set  $S$  with  $\varphi_{\frac{1}{2}}(S)$  satisfying the guarantee in Theorem 3, without the self-loop assumption. See Subsection 3.1.

Perhaps more interestingly, the same arguments can be used to prove that the Houdré-Tetali conjecture is true if we replace  $\frac{1}{2}$  by any constant  $p > \frac{1}{2}$ .

► **Theorem 5.** *Let  $(V, P, \pi)$  be an irreducible and reversible Markov chain. For any  $p \in (\frac{1}{2}, 1]$ ,*

$$(\varphi_p(P))^2 \leq \frac{4}{2p - 1} \cdot \lambda_2(I - P).$$

Similar to the discussion after Conjecture 2, this shows that the tight examples of the hard direction of Cheeger’s inequality must satisfy  $\varphi_{\frac{1}{2}+\epsilon}(G) \lesssim \sqrt{\frac{1}{\epsilon}} \cdot \varphi_1(G)$  for any  $\epsilon \in (0, \frac{1}{2})$ . Also, this provides an improved analysis of Cheeger’s inequality that if  $\varphi_1(G) \ll \sqrt{2p - 1} \cdot \varphi_p(G)$  then  $\varphi_1(G) \ll \sqrt{\lambda_2}$ . So this result has similar consequences as if Houdré and Tetali’s conjecture was true.

Finally, we observe that the same statement as in Theorem 5 can also be proved for non-reversible Markov chains, by replacing  $\lambda_2(I - P)$  with the eigenvalue defined for directed graphs by Chung [5]. See Theorem 9.

► **Remark 6** ( $\varphi_p$  for  $p < \frac{1}{2}$ ). For  $p < \frac{1}{2}$ , a simple argument shows that an inequality of the form in Theorem 5 cannot hold. To see it, consider the transformation  $P \rightarrow (1 - \delta)I + \delta P$  for some parameter  $0 < \delta < 1$  (equivalent to adding a large self loop when  $\delta$  is small) will scale  $\varphi_p$  by a factor  $\delta^p$ , while the second eigenvalue scales by a factor of  $\delta$ , and so the ratio  $\varphi_p(G)/\sqrt{\lambda_2(I - P_G)}$  scales by  $\delta^{p-\frac{1}{2}} \rightarrow \infty$  as  $\delta \rightarrow 0$ . When  $p = \frac{1}{2}$ , adding self loops does not change the ratio. Thus, it is the first exponent where such an inequality makes sense.

### 1.2 Previous Work on Boolean Hypercubes

The isoperimetric constant  $\varphi_{\frac{1}{2}}$  was initially studied by Talagrand in the Boolean hypercubes. Let  $\{0, 1\}^n$  be the  $n$ -dimensional hypercube. For a point  $x \in \{0, 1\}^n$ , let  $x^{\oplus i}$  be the point obtained by flipping the  $i$ -th bit of  $x$ . For a subset  $S \subset \{0, 1\}^n$ , if  $x \notin S$  define  $h_S(x) = 0$  and otherwise if  $x \in S$  define

$$h_S(x) := |\{i \in [n] \mid x^{\oplus i} \notin S\}|,$$

so that  $\sum_x h_S(x)$  is the size of the edge boundary of  $S$ . Let  $\mu$  be the uniform distribution on  $\{0, 1\}^n$  and  $\mu(S) := \sum_{x \in S} \mu(x)$ . The classical Poincaré inequality can be stated as, for any  $S \subset \{0, 1\}^n$ ,

$$\mu(S) \cdot (1 - \mu(S)) \lesssim \mathbb{E}_{x \sim \mu} [h_S(x)]. \tag{1}$$

## 36:4 On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs

Talagrand [20] proved a strengthening of the Poincaré inequality: For any  $S \subset \{0, 1\}^n$ ,

$$\mu(S) \cdot (1 - \mu(S)) \lesssim \mathbb{E}_{x \sim \mu} [\sqrt{h_S(x)}]. \quad (2)$$

The quantity  $\mathbb{E}\sqrt{h_S}$  is always smaller than  $\mathbb{E}h_S$  and can be seen as a different measure of the boundary information of  $S$ . Let  $\partial S := \{x \mid h_S(x) > 0\}$  be the vertex boundary of  $S$ . By the Cauchy-Schwarz inequality, Talagrand's theorem implies Margulis' theorem [17] that

$$\mu(S)^2 \cdot (1 - \mu(S))^2 \lesssim \mathbb{E}_{x \sim \mu} [h_S(x)] \cdot \mu(\partial S),$$

which was an original motivation for Talagrand to consider the quantity  $\mathbb{E}\sqrt{h_S}$ . More recently, both Margulis' and Talagrand's theorems inspired the analogs for directed graphs developed in [3, 9], to make major progresses in analyzing sublinear time algorithms for testing monotone functions. See also [6, 7] for a proof of a Talagrand's conjecture that further sharpens these inequalities.

The following remark clarifies the connection between  $\varphi_p$  and the quantities appearing in Poincaré's inequality and Talagrand's inequality.

► **Remark 7 ( $\varphi_p$  for Hypercubes).** For the  $n$ -dimensional Boolean hypercube  $Q_n$ , the stationary distribution  $\pi$  is simply the uniform distribution  $\mu$ . Note that the numerator in  $\varphi_1(Q_n)$  is exactly  $\frac{1}{n} \mathbb{E}_{x \in \mu} [h_f(x)]$ , and the Poincaré inequality translates to  $\varphi_1(Q_n) \gtrsim \frac{1}{n}$ . Similarly, the numerator of  $\varphi_{\frac{1}{2}}(Q_n)$  is exactly  $\frac{1}{\sqrt{n}} \mathbb{E}_{x \in \mu} [\sqrt{h_f(x)}]$ , and the Talagrand's inequality translates to  $\varphi_{\frac{1}{2}}(Q_n) \gtrsim \frac{1}{\sqrt{n}}$ .

Finally, we note that a parameter similar to  $\varphi_p$ , called  $h_f^p$ , was also studied in [6].

## 2 Preliminaries

Given two functions  $f, g$ , we use  $f \lesssim g$  to denote the existence of a positive constant  $c > 0$ , such that  $f \leq c \cdot g$  always holds. We use  $f \asymp g$  to denote  $f \lesssim g$  and  $g \lesssim f$ . For positive integers  $k$ , we use  $[k]$  to denote the set  $\{1, 2, \dots, k\}$ . For a function  $f : X \rightarrow \mathbb{R}$ ,  $\text{supp}(f)$  denotes the domain subset on which  $f$  is nonzero. The function  $\log x$  refers to the base  $e$  logarithm.

**Undirected Graphs.** Let  $G = (V, E)$  be an undirected graph. Let  $w : E \rightarrow \mathbb{R}_{\geq 0}$  be a weight function on the edges. The weighted degree of a vertex  $v$  is defined as  $\deg_w(v) := \sum_{e: e \ni v} w(e)$ . Let  $S \subset V$  be a nonempty subset of vertices. The edge boundary of  $S$  is defined as  $\delta(S) := \{e \in E \mid e \cap S \neq \emptyset \text{ and } e \cap \bar{S} \neq \emptyset\}$  and  $w(\delta(S))$  be the total edge weight of  $\delta(S)$ . The volume of  $S$  is defined as  $\text{vol}_w(S) := \sum_{v \in S} \deg_w(v)$ . The edge conductance of  $S$  and of  $G$  are defined as

$$\phi(S) := \frac{w(\delta(S))}{\text{vol}_w(S)} \quad \text{and} \quad \phi(G) := \min_{S: \text{vol}_w(S) \leq \text{vol}_w(V)/2} \phi(S).$$

In an undirected graph, the ordinary random walk has transition matrix  $P$  with  $P(u, v) = w(uv) / \deg_w(u)$  for every  $u, v \in V$ . If the graph is connected, then the stationary distribution  $\pi$  is unique with  $\pi(u) = \deg_w(u) / \sum_{v \in V} \deg_w(v)$  for every  $u \in V$ . It is thus straightforward to check that  $\phi(S) = \varphi_1(S)$  and  $\phi(G) = \varphi_1(G)$ , i.e. the two definitions coincide.

**Directed Graphs.** Let  $G = (V, E)$  be a directed graph. Let  $w : E \rightarrow \mathbb{R}_{\geq 0}$  be a weight function on the edges. The weighted indegree of a vertex  $v$  is defined as  $d_w^{\text{in}}(v) := \sum_{u:\vec{uv} \in E} w(\vec{uv})$  and the weighted outdegree of  $v$  is defined as  $d_w^{\text{out}}(v) := \sum_{u:\vec{vu} \in E} w(\vec{vu})$ . In a directed graph, the ordinary random walk has transition matrix  $P$  with  $P(u, v) = w(\vec{uv}) / \deg_w^{\text{out}}(u)$ . The stationary distribution  $\pi$  has no easy description but is unique as long as the directed graph is strongly connected. There are different notions of directed edge conductance for directed graphs. In analyzing random walks, the standard definition is exactly  $\varphi_1(G)$  as described in Definition 1, and this quantity is closely related to the mixing time of random walks; see e.g. [16, 5, 18]. In analyzing graph partitioning, there is a definition that extends the edge conductance above to directed graphs, which will not be used in this paper; see e.g. [21, 13].

**Spectral Graph Theory.** Given an undirected graph  $G = (V, E)$  with a weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$ , its adjacency matrix  $A = A(G)$  is an  $|V| \times |V|$  matrix where the  $(u, v)$ -th entry is  $w(uv)$ . The Laplacian matrix is defined as  $L := D - A$ , where  $D := \text{diag}(\{\deg_w(v)\}_{v \in V})$  is the diagonal degree matrix. The normalized adjacency matrix is defined as  $\mathcal{A} := D^{-1/2} A D^{-1/2}$ , and the normalized Laplacian matrix is defined as  $\mathcal{L} := I - \mathcal{A}$ . Let  $\lambda_1(\mathcal{L}) \leq \lambda_2(\mathcal{L}) \leq \dots \leq \lambda_n(\mathcal{L})$  be the eigenvalues of  $\mathcal{L}$ . It is known that  $\lambda_1(\mathcal{L}) = 0$  with eigenvector  $D^{1/2} \vec{1}$ , and

$$\lambda_2(\mathcal{L}) = \min_{g \perp D^{1/2} \vec{1}} \frac{g^T \mathcal{L} g}{g^T g} = \min_{f \perp D \vec{1}} \frac{f^T L f}{f^T D f} = \min_{f \perp D \vec{1}} \frac{\sum_{uv \in E} w(uv) \cdot (f(u) - f(v))^2}{\sum_v \deg_w(v) \cdot f(v)^2}.$$

Cheeger's inequality [4, 2, 1] is a fundamental result in spectral graph theory that connects edge conductance of an undirected graph  $G = (V, E)$  to the second smallest eigenvalue of its normalized Laplacian matrix:

$$\frac{\lambda_2}{2} \leq \phi(G) \leq \sqrt{2\lambda_2}.$$

The random walk transition matrix  $P$  is similar to the normalized Laplacian matrix  $\mathcal{A}$ , and the matrix  $I - P$  is similar to the normalized Laplacian matrix  $\mathcal{L}$ . In particular,  $I - P$  enjoys the same spectral properties as  $\mathcal{L}$  with real eigenvalues and a quadratic form characterization of  $\lambda_2$  as above; see Lemma 8.

Chung [5] defined the Laplacian matrix of a directed graph and used it to prove an analog of Cheeger's inequality. These will be stated in Subsection 3.2.

### 3 Positive Results

To prove Theorem 5, we follow standard spectral arguments used in proving Cheeger-type inequalities, in Trevisan's style. First, we start with the second eigenvector  $f_2 : V \rightarrow \mathbb{R}$  and truncate it so that the  $\pi$  weight of its support is at most half while preserving its Rayleigh quotient. The proof of the following lemma is standard and we defer it to the end of this section.

► **Lemma 8.** *Let  $(V, P, \pi)$  be an irreducible and reversible Markov chain. Let  $f_2$  be an eigenvector associated to the second smallest eigenvalue of the matrix  $I - P$ , with  $\pi(\{v \mid f_2(v) > 0\}) \leq \frac{1}{2}$ . Define the truncated vector  $f$  such that  $f(v) := \max\{f_2(v), 0\}$  for all  $v \in V$ . Then*

$$\lambda_2(I - P) \geq \frac{\sum_{f(i) \geq f(j)} \pi(i) \cdot P(i, j) \cdot (f(i) - f(j))^2}{\sum_{i \in V} \pi(i) f(i)^2}.$$

## 36:6 On the Houdré-Totali Conjecture About an Isoperimetric Constant of Graphs

Then the plan is to prove that one of the level sets has small isoperimetric constant. We index the vertices by  $[n]$  and order them so that  $f(i) \leq f(j)$  for  $i \leq j$ . For any  $t \geq 0$ , define the level set  $S_t := \{i \in [n] \mid f(i)^2 > t\}$ . By the construction of  $f$ , it holds that  $\pi(S_t) \leq \frac{1}{2}$  for any  $t \geq 0$ , and so  $\varphi_p(P) \leq \min_{t \geq 0} \varphi_p(S_t)$ . For convenience, we rescale  $f$  so that  $\max_i f(i) = 1$ .

To prove that one of  $S_t$  has small isoperimetric constant, we choose a uniform random  $t \in [0, 1]$  and consider  $\varphi_p(S_t)$ . We will bound the expected value of the numerator of  $\varphi_p(S_t)$  and of the denominator of  $\varphi_p(S_t)$  and conclude that there exists a  $t$  with  $\varphi_p(S_t)$  at most the ratio of the expected values, i.e.  $\min_{t \geq 0} \varphi_p(S_t) \leq \mathbb{E}_t[\text{numerator of } \varphi_p(S_t)] / \mathbb{E}_t[\text{denominator of } \varphi_p(S_t)]$ .

The expected value of the denominator is easy to analyze. Since we choose  $t$  uniformly randomly, each vertex  $i$  is included in  $S_t$  with probability  $f(i)^2$ , and thus

$$\mathbb{E}_t[\pi(S_t)] = \sum_{i \in V} \pi(i) \cdot f(i)^2.$$

The rest of the proof is to analyze the expected value of the numerator  $\sum_{i \in V} \pi(i) \cdot P(i, \overline{S_t})^p$ . For a vertex  $i$ , if the random threshold  $t$  is between  $f(j)^2$  and  $f(j-1)^2$  with  $f(j) > f(j-1)$ , then  $P(i, \overline{S_t}) = P(i, [j-1])$ , and so

$$\begin{aligned} \mathbb{E}_t[P(i, \overline{S_t})^p] &= \sum_{j=1}^i (f(j)^2 - f(j-1)^2) \cdot P(i, [j-1])^p \\ &= \sum_{j=1}^i (f(j)^2 - f(j-1)^2) \cdot \sum_{l=1}^{j-1} (P(i, [l])^p - P(i, [l-1])^p) \\ &= \sum_{l=1}^{i-1} (f(i)^2 - f(l)^2) \cdot (P(i, [l])^p - P(i, [l-1])^p), \end{aligned}$$

where the second equality is by writing a telescoping sum and the third equality is by a change of summation. So, the expected numerator  $\mathbb{E}_t[\sum_{i=1}^n \pi(i) \cdot P(i, \overline{S_t})^p]$  is

$$\begin{aligned} &\sum_{i=1}^n \sum_{j=1}^{i-1} \pi(i) \cdot (f(i)^2 - f(j)^2) \cdot (P(i, [j])^p - P(i, [j-1])^p) \\ &\leq \sqrt{\sum_{i=1}^n \sum_{j=1}^{i-1} \pi(i) \cdot (f(i) - f(j))^2 \cdot P(i, j)} \\ &\quad \cdot \sqrt{\sum_{i=1}^n \sum_{j=1}^{i-1} \pi(i) \cdot (f(i) + f(j))^2 \cdot \frac{(P(i, [j])^p - P(i, [j-1])^p)^2}{P(i, j)}} \\ &\leq \sqrt{\lambda_2 \cdot \sum_{i=1}^n \pi(i) \cdot f(i)^2} \cdot \sqrt{\sum_{i=1}^n 4 \cdot \pi(i) \cdot f(i)^2 \cdot \sum_{j=1}^{i-1} \frac{(P(i, [j])^p - P(i, [j-1])^p)^2}{P(i, [j]) - P(i, [j-1])}}, \end{aligned}$$

where the first inequality is by Cauchy-Schwarz, and the second inequality is by Lemma 8 and  $(f(i) + f(j))^2 \leq 4f(i)^2$  and  $P(i, j) = P(i, [j]) - P(i, [j-1])$ .

To upper bound the inner sum of the second term, we denote  $a_j := P(i, [j])$  and it suffices to upper bound the sum of the form  $\sum_{j=1}^n (a_j^p - a_{j-1}^p)^2 / (a_j - a_{j-1})$  with  $a_0 = 0$  and  $a_n \leq 1$ , with a bound independent of  $n$ . Let  $C(n, a)$  denote the supremum of the sum when  $a_n = a$ . Note that  $C(n, a) = a^{2p-1} \cdot C(n, 1)$  by a simple scaling argument. Let  $(a_i)_{i=1}^n$  be an optimal sequence that achieves the supremum of  $C(n, 1)$ . Then,

$$\begin{aligned} C(n, 1) &= C(n-1, a_{n-1}) + \frac{(1-a_{n-1}^p)^2}{1-a_{n-1}} = a_{n-1}^{2p-1} \cdot C(n-1, 1) + \frac{(1-a_{n-1}^p)^2}{1-a_{n-1}} \\ &\leq a_{n-1}^{2p-1} \cdot C(n, 1) + \frac{(1-a_{n-1}^p)^2}{1-a_{n-1}}. \end{aligned}$$

It follows that

$$C(n, 1) \leq \sup_{a \in [0, 1]} \frac{(1-a^p)^2}{(1-a)(1-a^{2p-1})} \leq \sup_{a \in [0, 1]} \frac{(1-a^p)^2}{(2p-1)(1-a)^2} \leq \frac{1}{2p-1},$$

where the second inequality is by the mean value theorem that  $1-a^{2p-1} \geq (2p-1)(1-a)$  and the last inequality is because  $a \in [0, 1]$  and  $p \in (\frac{1}{2}, 1]$ . Clearly,  $C(n, 1) \geq C(n, a_n)$  for any  $a_n \in [0, 1]$ , and so the inner sum of the second term in the expected numerator is at most  $\frac{1}{2p-1}$ . Putting together, this completes the proof of Theorem 5 as

$$\varphi_p(P) \leq \min_{t: t > 0} \varphi_p(S_t) \leq \frac{\mathbb{E}_t[\sum_{i=1}^n \pi(i) \cdot P(i, \overline{S_t})^p]}{\mathbb{E}_t[\pi(S_t)]} \leq 2\sqrt{\lambda_2} \sqrt{\frac{1}{2p-1}},$$

which implies that  $(\varphi_p(P))^2 \leq \frac{4}{2p-1} \cdot \lambda_2$ .

### 3.1 Recovering Morris and Peres's Result

To recover Theorem 3, we follow the same arguments but add a truncation step so that the sequence  $a_i$  above will start with  $a_0 \approx \varphi_{\frac{1}{2}}(P)$ . In this subsection, we plug in  $p = \frac{1}{2}$ . As above, the main work is to upper bound the expected value of the numerator. Recall that

$$\mathbb{E}_t \left[ \sqrt{P(i, \overline{S_t})} \right] = \sum_{j=1}^{i-1} (f(i)^2 - f(j)^2) \cdot (\sqrt{P(i, [j])} - \sqrt{P(i, [j-1])}),$$

Let  $l_i$  be the index such that  $\sqrt{P(i, [l_i])} \leq \frac{1}{2}\varphi_{\frac{1}{2}}(P)$  but  $\sqrt{P(i, [l_i+1])} > \frac{1}{2}\varphi_{\frac{1}{2}}(P)$ . Then, we can upper bound the right hand side by

$$\begin{aligned} \mathbb{E}_t \left[ \sqrt{P(i, \overline{S_t})} \right] &\leq \frac{1}{2}\varphi_{\frac{1}{2}}(P) \cdot f(i)^2 + (f(i)^2 - f(l_i+1)^2) \left( \sqrt{P(i, [l_i+1])} - \frac{1}{2}\varphi_{\frac{1}{2}}(P) \right) \\ &\quad + \sum_{j=l_i+2}^{i-1} (f(i)^2 - f(j)^2) \left( \sqrt{P(i, [j])} - \sqrt{P(i, [j-1])} \right). \end{aligned}$$

To shorten the expression, let us use the notations

$$a_{i,0} = \frac{1}{2}\varphi_{\frac{1}{2}}(P) \quad \text{and} \quad a_{i,j} = \sqrt{P(i, [l_i+j])}.$$

Summing over  $i$  and using these notations, the expected numerator is

$$\begin{aligned} \mathbb{E}_t \left[ \sum_{i=1}^n \pi(i) \cdot \sqrt{P(i, \overline{S_t})} \right] &\leq \frac{1}{2}\varphi_{\frac{1}{2}}(P) \cdot \sum_{i=1}^n \pi(i) \cdot f(i)^2 \\ &\quad + \underbrace{\sum_{i=1}^n \pi(i) \cdot \sum_{j=1}^{i-l_i-1} (f(i)^2 - f(l_i+j)^2) \cdot (a_{i,j} - a_{i,j-1})}_{(*)} \end{aligned}$$

Applying Cauchy-Schwarz as before gives

$$\begin{aligned}
 (*) &\leq \sqrt{\sum_{i=1}^n \sum_{j=1}^{i-l_i-1} \pi(i) \cdot (f(i) - f(l_i + j))^2 \cdot P(i, l_i + j)} \\
 &\quad \cdot \sqrt{\sum_{i=1}^n \sum_{j=1}^{i-l_i-1} \pi(i) \cdot (f(i) + f(l_i + j))^2 \cdot \frac{(a_{i,j} - a_{i,j-1})^2}{P(i, l_i + j)}} \\
 &\leq \sqrt{\lambda_2 \cdot \sum_{i=1}^n \pi(i) \cdot f(i)^2} \cdot \sqrt{\sum_{i=1}^n 4 \cdot \pi(i) \cdot f(i)^2 \cdot \underbrace{\sum_{j=1}^{i-1} \frac{a_{i,j} - a_{i,j-1}}{a_{i,j} + a_{i,j-1}}}_{(**)}},
 \end{aligned}$$

where the second inequality uses Lemma 8 and  $P(i, l_i + j) = a_{i,j}^2 - a_{i,j-1}^2$ .

To upper bound (\*\*), we let  $b_i := a_{i,j}$  and use that  $\frac{1}{2}\varphi_{\frac{1}{2}}(P) = b_0 \leq b_1 \leq \dots \leq b_m \leq 1 =: b_{m+1}$  to upper bound the function

$$f : (b_0, b_1, \dots, b_m) \rightarrow \frac{b_1 - b_0}{b_1 + b_0} + \frac{b_2 - b_1}{b_2 + b_1} + \dots + \frac{b_m - b_{m-1}}{b_m + b_{m-1}} + \frac{1 - b_m}{1 + b_m}$$

The partial derivative of  $f$  is

$$\frac{\partial f}{\partial b_i} = \frac{2b_{i-1}}{(b_i + b_{i-1})^2} - \frac{2b_{i+1}}{(b_{i+1} + b_i)^2} = \frac{2(b_{i+1} - b_{i-1})(b_{i-1}b_{i+1} - b_i^2)}{(b_i + b_{i-1})^2(b_{i+1} + b_i)^2}.$$

Since  $b_{i+1} - b_{i-1} > 0$  by definition, the function increases up until  $b_i^2 = b_{i-1}b_{i+1}$  and then decreases. So, the maximum is attained when  $b_i = (b_0)^{\frac{m+1-i}{m+1}}$  with  $b_0 = \frac{1}{2}\varphi_{\frac{1}{2}}(P)$  and  $b_{m+1} = 1$ , in which case the sum is

$$\sum_{i=1}^{m+1} \frac{b_i - b_{i-1}}{b_i + b_{i-1}} = \sum_{i=1}^{m+1} \frac{1 - \frac{b_{i-1}}{b_i}}{1 + \frac{b_{i-1}}{b_i}} = \sum_{i=1}^{m+1} \frac{1 - b_0^{\frac{1}{m+1}}}{1 + b_0^{\frac{1}{m+1}}} = (m+1) \cdot \frac{1 - b_0^{\frac{1}{m+1}}}{1 + b_0^{\frac{1}{m+1}}}$$

For  $b_0 \in [0, 1]$ , this value is increasing when  $m$  increases, and so the sum is upper bounded by

$$\begin{aligned}
 (**)&\leq \lim_{x \rightarrow \infty} x \cdot \frac{1 - b_0^{\frac{1}{x}}}{1 + b_0^{\frac{1}{x}}} = \lim_{x \rightarrow \infty} x \cdot \frac{1 - b_0^{\frac{1}{x}}}{2} = \lim_{y \rightarrow 0} \frac{1 - b_0^y}{2y} \\
 &= \lim_{y \rightarrow 0} \frac{-b_0^y \log b_0}{2} = \frac{1}{2} \log \frac{1}{b_0} = \frac{1}{2} \log \frac{2}{\varphi_{\frac{1}{2}}(P)},
 \end{aligned}$$

where the third last equality is by L'Hôpital's rule. Plugging this back into (\*\*) and (\*), the expected numerator is

$$\begin{aligned}
 \mathbb{E}_t \left[ \sum_{i=1}^n \pi(i) \cdot \sqrt{P(i, S_t)} \right] &\leq \frac{1}{2} \varphi_{\frac{1}{2}}(P) \cdot \sum_{i=1}^n \pi(i) \cdot f(i)^2 \\
 &\quad + \sqrt{\lambda_2 \cdot \sum_{i=1}^n \pi(i) \cdot f(i)^2} \cdot \sqrt{\sum_{i=1}^n 2 \log \frac{2}{\varphi_{\frac{1}{2}}(P)} \cdot \pi(i) \cdot f(i)^2}.
 \end{aligned}$$

As before, the expected denominator is  $\mathbb{E}_t[\pi(S_t)] = \sum_{i \in V} \pi(i) \cdot f(i)^2$ . Putting together,

$$\varphi_{\frac{1}{2}}(P) \leq \min_{t:t>0} \varphi_{\frac{1}{2}}(S_t) \leq \frac{\mathbb{E}_t \left[ \sum_{i=1}^n \pi(i) \cdot \sqrt{P(i, S_t)} \right]}{\mathbb{E}_t[\pi(S_t)]} \leq \frac{1}{2} \varphi_{\frac{1}{2}}(P) + \sqrt{2 \log \frac{2}{\varphi_{\frac{1}{2}}(P)}} \lambda_2.$$

Rearranging recovers Theorem 3.



### 3.2 Non-Reversible Markov Chains

Houdré and Tetali only formulated Conjecture 2 for reversible Markov chains of which the eigenvalues of  $I - P$  are real. For non-reversible Markov chains, we observe that Chung's definition of eigenvalues for directed graphs [5] can be used to obtain the same results in Theorem 3 and Theorem 5.

Given a directed graph  $G = (V, E)$  with a weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$ , let  $P_G$  be the transition matrix of the ordinary random walks on  $G$  with  $P_G(u, v) = w(uv) / \sum_{v \in V} w(uv)$  for each edge  $uv \in E$ . Suppose  $G$  is strongly connected, then there is a unique stationary distribution  $\pi : V \rightarrow \mathbb{R}_+$  such that  $\pi^T P = \pi^T$ . Let  $\Pi = \text{diag}(\pi)$ . Chung defined the Laplacian of the directed graph  $G$  as

$$\vec{L}_G := I - \frac{1}{2} \left( \Pi^{\frac{1}{2}} P \Pi^{-\frac{1}{2}} + \Pi^{-\frac{1}{2}} P^T \Pi^{\frac{1}{2}} \right).$$

Since  $\vec{L}_G$  is a real symmetric matrix, its eigenvalues are real. Let  $\lambda_2$  be the second smallest eigenvalue of  $\vec{L}_G$ . Chung [5] proved an analog of Cheeger's inequality that

$$\frac{1}{2} \varphi_1(G)^2 \leq \lambda_2(\vec{L}_G) \leq 2 \varphi_1(G).$$

We observe that  $\lambda_2(\vec{L}_G)$  can be used to extend our results to non-reversible Markov chains.

► **Theorem 9.** *Let  $(V, P, \pi)$  be an irreducible Markov chain. For any  $p \in (\frac{1}{2}, 1]$ ,*

$$(\varphi_p(P))^2 \leq \frac{4}{2p-1} \cdot \lambda_2(\vec{L}_G).$$

For  $p = 1/2$ ,

$$\lambda_2(\vec{L}_G) \gtrsim \frac{(\varphi_{\frac{1}{2}}(P))^2}{\log(1/\varphi_{\frac{1}{2}}(P))}.$$

Note that the main proofs of Theorem 5 and Theorem 3 (i.e. computing the expected numerator) did not require the Markov chain to be reversible. The reversible assumption was only used in characterizing the second eigenvalue in Lemma 8. The following is an analog of Lemma 8 for non-reversible Markov chains using Chung's definition of the second eigenvalue of directed graphs.

► **Lemma 10.** *Let  $(V, P, \pi)$  be an irreducible Markov chain. Let  $v_2$  be an eigenvector associated to the second smallest eigenvalue of the matrix  $\vec{L}_G$ . Define the reweighted eigenvector  $f_2 := \Pi^{-\frac{1}{2}} v_2$ , with  $\pi(\{v : f_2(v) \geq 0\}) \leq \frac{1}{2}$ . Define the truncated vector  $f := \max(f_2, 0)$ . Then*

$$\lambda_2(\vec{L}_G) \geq \frac{\sum_{u, v \in V: f(u) \geq f(v)} \pi(u) \cdot P(u, v) \cdot (f(u) - f(v))^2}{\sum_{v \in V} \pi(v) f(v)^2}.$$

With this lemma, we can follow the proofs of Theorem 5 and Theorem 3 verbatim as in above, by defining level sets  $S_t$  using the truncated vector  $f$  and computing the expected numerator and denominator and so on.

This concludes the proof of Theorem 9. We will prove Lemma 10 in the next subsection.

### 3.3 Proofs of Auxiliary Lemmas

In this subsection, we prove Lemma 8 and Lemma 10. The proofs are standard but we include them for completeness.

**Proof of Lemma 8.** For  $f, g : V \rightarrow \mathbb{R}$ , we define  $\langle f, g \rangle_\pi := \sum_{i \in V} \pi(i) \cdot f(i) \cdot g(i)$ . By definition of the second eigenvector,  $\langle (I - P)f_2, f_2 \rangle_\pi = \lambda_2 \langle f_2, f_2 \rangle_\pi$ .

For  $f := \max(f_2, 0)$ , note that  $Pf \geq Pf_2$ , as

$$(Pf)(i) = \sum_{j \in V} p(i, j) f(j) \geq \sum_{j \in V} p(i, j) f_2(j) = (Pf_2)(i),$$

and thus

$$\langle Pf, f \rangle_\pi = \sum_{i \in V: f_2(i) \geq 0} \pi(i) \cdot (Pf)(i) \cdot f(i) \geq \sum_{i \in V: f_2(i) \geq 0} \pi(i) \cdot (Pf_2)(i) \cdot f_2(i) = (1 - \lambda_2) \langle f, f \rangle_\pi,$$

where the last equality uses that  $Pf_2 = (1 - \lambda_2)f_2$ . It follows that

$$\lambda_2 \geq \frac{\langle (I - P)f, f \rangle_\pi}{\langle f, f \rangle_\pi}.$$

The denominator is the same as the denominator in the statement. It remains to check that the numerator is also the same as the numerator in the statement. By direct calculation,

$$\begin{aligned} \langle (I - P)f, f \rangle_\pi &= \sum_{i \in V} \pi(i) f(i)^2 - \sum_{i \in V} \pi(i) \sum_{j \in V} P(i, j) f(j) f(i) \\ &= \sum_{i \in V} \sum_{j \in V} \pi(i) P(i, j) f(i)^2 - \sum_{i \in V} \sum_{j \in V} \pi(i) P(j, i) f(j) f(i) \\ &= \sum_{i \in V} \sum_{j \in V} \pi(i) P(i, j) \left( \frac{1}{2} (f(i)^2 + f(j)^2) - f(i) f(j) \right) \\ &= \sum_{i > j} \pi(i) P(i, j) (f(i) - f(j))^2, \end{aligned}$$

where the second equality uses  $\sum_{j \in V} P(i, j) = 1$  and the third equality uses reversibility which gives  $\pi(i) P(i, j) = \pi(j) P(j, i)$  for all  $i, j \in V$ , to get  $\sum_{i, j} \pi(i) P(i, j) f(i)^2 = \sum_{i, j} \pi(i) P(i, j) f(j)^2$ .  $\blacktriangleleft$

To prove Lemma 10, we will use the following facts about  $\lambda_2(\vec{L}_G)$  in [5].

► **Lemma 11 ([5]).** *Let  $G = (V, E)$  be a strongly connected directed graph and  $\pi$  be its stationary distribution. The second smallest eigenvalue  $\lambda_2$  of the directed Laplacian  $\vec{L}_G$  satisfies*

$$\lambda_2 = \inf_{f \perp \pi} \frac{\sum_{u, v \in V} \pi(u) \cdot P(u, v) \cdot |f(u) - f(v)|^2}{\sum_{v \in V} \pi(v) \cdot |f(v)|^2}$$

Suppose  $v_2$  is an eigenvector of  $\vec{L}_G$  associated with eigenvalue  $\lambda_2$ . Then, for the reweighted eigenvector  $f_2 := \Pi^{-\frac{1}{2}} v_2$ , for all  $u \in V$ ,

$$\lambda_2 \cdot f_2(u) \cdot \pi(u) = \frac{1}{2} \sum_v (f_2(u) - f_2(v)) \cdot (\pi(u) P(u, v) + \pi(v) P(v, u)).$$

**Proof of Lemma 10.** We claim that the truncated vector  $f := \max\{f_2, 0\}$  satisfies

$$\lambda_2 \cdot f(u) \cdot \pi(u) \geq \frac{1}{2} \sum_v (f(u) - f(v)) \cdot (\pi(u)P(u, v) + \pi(v)P(v, u)).$$

for all  $u \in V$ . Indeed, for  $u$  such that  $f(u) > 0$ ,

$$\begin{aligned} \lambda_2 \cdot f(u) \cdot \pi(u) &= \lambda_2 \cdot f_2(u) \cdot \pi(u) \\ &= \frac{1}{2} \sum_{v \in V} (f_2(u) - f_2(v)) \cdot (\pi(u)P(u, v) + \pi(v)P(v, u)) \\ &\geq \frac{1}{2} \sum_{v \in V} (f(u) - f(v)) \cdot (\pi(u)P(u, v) + \pi(v)P(v, u)), \end{aligned}$$

where the second equality is by the fact above and the last inequality is by  $f_2(u) - f_2(v) \geq f(u) - f(v)$  for all  $u, v \in V$  due to truncation. For  $u$  such that  $f(u) = 0$ , the inequality holds trivially because

$$\lambda_2 \cdot f(u) \cdot \pi(u) = 0 \geq \frac{1}{2} \sum_v (-f(v)) \cdot (\pi(u)P(u, v) + \pi(v)P(v, u))$$

as  $f(v) \geq 0$  for all  $v$  by truncation. Thus the claim follows. Multiplying both sides of the claim by  $f(u)$  and then summing over all  $u$  gives

$$\begin{aligned} \lambda_2 \cdot \sum_{u \in V} f^2(u) \pi(u) &\geq \frac{1}{2} \sum_{u \in V} f(u) \sum_{v \in V} (f(u) - f(v)) \cdot (\pi(u)P(u, v) + \pi(v)P(v, u)) \\ &= \frac{1}{2} \sum_{u \in V} \sum_{v \in V} \pi(u) \cdot P(u, v) \cdot \left( \frac{1}{2} f(u)^2 + \frac{1}{2} f(v)^2 - f(u)f(v) \right) \\ &= \frac{1}{2} \sum_{u \in V} \sum_{v \in V} \pi(u) \cdot P(u, v) \cdot (f(u) - f(v))^2. \end{aligned}$$

This is equivalent to the statement where the sum is over pairs with  $f(u) \geq f(v)$ . ◀

## 4 Counterexamples

In this section, we prove Theorem 4 by constructing a family of counterexamples and bounding their second eigenvalues and  $\varphi_{\frac{1}{2}}$  value. The construction is simple.

► **Definition 12 (Counterexamples).** Let  $G_n$  be a graph with vertex set  $[n]$ . For each  $i, j \in [n], i \neq j$ , the edge weight is

$$P(i, j) = \frac{1}{C(\min\{|i - j|, n - |i - j|\})^3},$$

where  $C = \sum_{i=1}^n 1/\min\{|i - j|, n - |i - j|\}^3$  is the normalizing constant to make the graph 1-regular.

We will prove the two claims in Theorem 4 about the second smallest eigenvalue and the  $\varphi_{\frac{1}{2}}(G)$  value. First, we analyze the second smallest eigenvalue, based on the construction that  $I - P$  is a circulant matrix.

► **Lemma 13.** For  $G_n$  in Definition 12, the second smallest eigenvalue of  $I - P$  is

$$\lambda_2(I - P) \lesssim \frac{\log n}{n^2}.$$

## 36:12 On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs

**Proof.** By our construction, the graph  $G_n$  is cyclic that  $P(i, j) = P((i+k) \bmod n, (j+k) \bmod n)$  for all  $i, j, k \in [n]$ . So the matrix  $I - P$  is a circulant matrix of the form

$$I - P = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_0 & \dots & \dots & a_{n-2} \\ a_{n-2} & a_{n-1} & \ddots & \ddots & a_{n-3} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_1 & a_2 & \dots & \dots & a_0 \end{pmatrix}$$

where  $a_0 = 1$  and  $a_j = -P(1, j+1)$  for all  $j \in [n]$ . It is well-known that an  $n \times n$  circulant matrix with first row entries  $a \in \mathbb{R}^n$  has eigenvalues and corresponding eigenvectors

$$\left\{ \sum_{i=0}^{n-1} a_i \omega_k^i \right\}_{k=0}^{n-1} \quad \text{and} \quad \left\{ (1, \omega_k, \dots, \omega_k^{n-1})^T \right\}_{k=0}^{n-1}$$

where  $\omega_k := e^{\frac{2\pi k i}{n}}$  are the  $n$ -th roots of unity for  $k \in [n]$  (where  $i$  denotes the imaginary number).

So, the second smallest eigenvalue  $\lambda_2$  of  $I - P$  corresponds to the first  $n$ -th root of unity  $\omega := \omega_1 = e^{\frac{2\pi i}{n}}$ , and

$$\lambda_2 = \sum_{i=0}^{n-1} a_i \cdot \omega^i = \sum_{i=2}^n P(1, i) - \sum_{i=2}^{\lfloor n/2 \rfloor + 1} P(1, i) \cdot \omega^{i-1} - \sum_{i=\lfloor n/2 \rfloor + 2}^n P(1, i) \cdot \omega^{i-1}.$$

We consider two cases, when  $n$  is odd and  $n$  is even. When  $n = 2k + 1$  is odd, note that by definition  $P(1, i) = P(1, 2k + 3 - i)$  for  $2 \leq i \leq k + 1$  and so we can pair up the terms in the above equation to get

$$\lambda_2 = \sum_{i=2}^{k+1} P(1, i) \left( 2 - \omega^{i-1} - \frac{1}{\omega^{i-1}} \right) = - \sum_{i=2}^{k+1} P(1, i) \left( \omega^{\frac{i-1}{2}} - \frac{1}{\omega^{\frac{i-1}{2}}} \right)^2.$$

Using the definition of  $\omega^k := e^{\frac{2\pi k i}{n}} = \cos \frac{2k\pi}{n} + i \sin \frac{2k\pi}{n}$ , it follows that

$$\begin{aligned} \lambda_2 &= - \sum_{i=2}^{k+1} P(1, i) \left( \omega^{\frac{i-1}{2}} - \frac{1}{\omega^{\frac{i-1}{2}}} \right)^2 = - \sum_{i=2}^{k+1} P(1, i) \left( 2i \sin \frac{(i-1)\pi}{n} \right)^2 \\ &= 4 \sum_{i=2}^{k+1} P(1, i) \left( \sin \frac{(i-1)\pi}{n} \right)^2. \end{aligned}$$

Finally we use the fact that  $\sin x < x$  and that  $P(1, i) = \frac{1}{C \cdot \min\{|i-1|, n-|i-1|\}^3} < \frac{1}{(i-1)^3}$  for  $2 \leq i \leq k + 1$  as  $C \geq 1$  to conclude that

$$\lambda_2 < 4 \sum_{i=2}^{k+1} \frac{1}{(i-1)^3} \left( \frac{(i-1)\pi}{n} \right)^2 = 4 \sum_{i=1}^k \frac{1}{i} \left( \frac{\pi^2}{n^2} \right) \lesssim \frac{\log n}{n^2}.$$

When  $n = 2k$  is even, the proof follows along the same lines, but we need to remove the term  $k = \frac{n}{2} + 1$  in the sum because  $\omega^{n/2} = -1$ . However, it only contributes a term of  $O\left(\frac{1}{n^3}\right)$  to the sum, which is negligible.  $\blacktriangleleft$

It remains to prove that  $\varphi_{\frac{1}{2}}(P) \gtrsim \frac{\log n}{n}$ . As this graph is symmetric, our intuition is that  $\varphi_{\frac{1}{2}}(P)$  attains its minimum at the set  $S = \{1, \dots, \frac{n}{2}\}$ . In this case, for each vertex  $i \in S$ ,

$$\sqrt{P(i, \bar{S})} \geq \sqrt{\sum_{j=i}^{\frac{i+n}{2}} \frac{1}{j^3}} \approx \frac{1}{i} - \frac{1}{i + \frac{n}{2}} \geq \frac{1}{2i},$$

which implies that

$$\varphi_{\frac{1}{2}}(S) \gtrsim \sum_{i=1}^{\frac{n}{2}} \frac{1}{n} \sqrt{P(i, \bar{S})} \gtrsim \frac{1}{n} \sum_{i=1}^{\frac{n}{2}} \frac{1}{i} \gtrsim \frac{\log n}{n}.$$

Our plan was to prove that  $S$  indeed attains the minimum, but we do not have such a proof. Instead, we will work on a slightly different lower bound, which satisfies a concavity property that allows us to argue that sets of consecutive vertices attain the minimum, in order to prove the lower bound. It turns out that the proof is a bit long and we will present it in the next subsection.

#### 4.1 Proof of $\varphi_{\frac{1}{2}}$ Lower Bound

First, we set up some notations for the proof. Let us partition the vertex set of  $G_n$  into two sets  $A$  and  $B := G \setminus A$  with  $|A| \leq |B|$ . As the graph  $G_n$  is cyclic, we can arrange the vertices  $V = [n]$  in a clockwise manner and without loss of generality we assume  $1 \in A$  and  $n \in B$ . Let us divide the vertices of  $A$  and  $B$  into contiguous sets  $A_1, B_1, A_2, B_2, \dots, A_k, B_k$  in the cyclic representation, and denote their sizes by  $a_i := |A_i|$  and  $b_i := |B_i|$  for  $1 \leq i \leq k$ . More explicitly, for  $1 \leq i \leq k$ , the vertices in  $A_i$  and  $B_i$  are

$$A_i = \left\{ \sum_{j=1}^{i-1} a_j + \sum_{j=1}^{i-1} b_j + 1, \dots, \sum_{j=1}^i a_j + \sum_{j=1}^{i-1} b_j \right\}, B_i = \left\{ \sum_{j=1}^i a_j + \sum_{j=1}^{i-1} b_j + 1, \dots, \sum_{j=1}^i a_j + \sum_{j=1}^{i-1} b_j \right\}.$$

For two disjoint subsets  $S, T \subset V$ , let us define  $f(S, T) := \sum_{u \in S} \sqrt{P(u, T)}$ . Note that  $\varphi_{\frac{1}{2}}(A) = \frac{f(A, B)}{|A|}$ , so our goal is to lower bound

$$f(A, B) = \sum_{i=1}^k f(A_i, B).$$

For two sets  $S, T \in \{A_i\}_{i=1}^k \cup \{B_i\}_{i=1}^k$ , let us define the contiguous block  $[S, T]$  to be the block of sets from  $S$  clockwise up until  $T$ , possibly going around. For example,  $[B_k, A_2] := B_k \cup A_1 \cup B_1 \cup A_2$ , and note that  $[S, T] \neq [T, S]$  since the sets are counted clockwise.

After we set up the notations, we start with a lower bound on  $f(A_i, B)$  by a natural function, the logarithm of the size of contiguous sets, which is the ‘‘slightly different lower bound’’ that we mentioned before this subsection.

► **Lemma 14.** For  $1 \leq i \leq k$ ,

$$\begin{aligned} \sqrt{2C} \cdot f(A_i, B) \geq & \sum_{j=1}^k \left( \log(|[A_i, A_j]| + 1) + \log(|[B_i, B_{j-1}]| + 1) \right. \\ & \left. - \log(|[B_i, A_j]| + 1) - \log(|[A_i, B_{j-1}]| + 1) \right), \end{aligned}$$

where  $C$  is the normalizing constant in Definition 12 and  $|[S, T]|$  denotes the number of vertices in the block  $[S, T]$ .

**Proof.** We prove the statement for  $f(A_1, B)$ . By definition of  $f, A_i, B_i$  stated above,

$$\begin{aligned} \sqrt{2C} \cdot f(A_1, B) &= \sqrt{2C} \cdot \sum_{i \in A_1} \sqrt{\sum_{l=1}^k P(i, B_l)} \\ &= \sqrt{2C} \cdot \sum_{i=1}^{a_1} \sqrt{\sum_{l=1}^k \sum_{j=1}^{b_l} P\left(i, \sum_{m=1}^l a_m + \sum_{m=1}^{l-1} b_m + j\right)}. \end{aligned}$$

### 36:14 On the Houdré-Totali Conjecture About an Isoperimetric Constant of Graphs

By the definition of  $P$  in Definition 12,  $P(i, j) \geq \frac{1}{C|i-j|^3}$  and so

$$\begin{aligned} \sqrt{2C} \cdot f(A_1, B) &\geq \sqrt{2} \cdot \sum_{i=1}^{a_1} \sqrt{\sum_{l=1}^k \sum_{j=1}^{b_l} \left( \sum_{m=1}^l a_m + \sum_{m=1}^{l-1} b_m + j - i \right)^{-3}} \\ &= \sqrt{2} \cdot \sum_{i=0}^{a_1-1} \sqrt{\sum_{l=1}^k \sum_{j=1}^{b_l} \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + j + i \right)^{-3}}. \end{aligned}$$

We lower bound the inner sum by an integral, so that

$$\begin{aligned} \sqrt{2C} \cdot f(A_1, B) &\geq \sqrt{2} \cdot \sum_{i=0}^{a_1-1} \sqrt{\sum_{l=1}^k \int_1^{b_l+1} \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + x + i \right)^{-3} dx} \\ &= \sum_{i=0}^{a_1-1} \left( \sum_{l=1}^k \underbrace{\left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + i + 1 \right)^{-2}}_{\alpha_l} \right. \\ &\quad \left. - \underbrace{\left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + i + 1 \right)^{-2}}_{\beta_l} \right)^{1/2}. \end{aligned}$$

Now we use the following simple inequality about decreasing numbers.

▷ **Claim 15.** Let  $(\alpha_i)_{i=1}^k, (\beta_i)_{i=1}^k$  be positive real numbers such that  $\alpha_1 \geq \beta_1 \geq \alpha_2 \geq \beta_2 \geq \dots \geq \alpha_k \geq \beta_k \geq 0$ . Then

$$\sum_{i=1}^k (\alpha_i^2 - \beta_i^2) \geq \left( \sum_{i=1}^k (\alpha_i - \beta_i) \right)^2.$$

*Proof.* The proof is by induction. For  $i = 1$ , the claim is clear as  $\alpha_1^2 - \beta_1^2 \geq (\alpha_1 - \beta_1)^2$ . Suppose that the claim is true for  $i = k$ . Let  $A = \sum_{i=2}^{k+1} (\alpha_i^2 - \beta_i^2)$  and  $B = \sum_{i=2}^{k+1} (\alpha_i - \beta_i)$ . For the induction step, we need to show that  $\alpha_1^2 - \beta_1^2 + A \geq (\alpha_1 - \beta_1 + B)^2$ . Since  $A \geq B^2$  by induction, it suffices to show that  $\alpha_1^2 - \beta_1^2 \geq (\alpha_1 - \beta_1)(\alpha_1 - \beta_1 + 2B)$ , which is equivalent to  $\beta_1 \geq B$ . It follows from the property of decreasing sequence that  $B \leq \alpha_2 \leq \beta_1$ , verifying the induction step. ◁

The  $\sqrt{\alpha_l}$  and  $\sqrt{\beta_l}$  in the right hand side above satisfy the assumptions of the claim, and thus

$$\sqrt{2C} \cdot f(A_1, B) \geq \sum_{l=1}^k \sum_{i=0}^{a_1-1} \left( \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + i + 1 \right)^{-1} - \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + i + 1 \right)^{-1} \right)$$

We again lower bound the inner sum by an integral so that  $\sqrt{2C} \cdot f(A_1, B)$  is at least

$$\begin{aligned} &\sum_{l=1}^k \int_0^{a_1} \left( \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + x + 1 \right)^{-1} - \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + x + 1 \right)^{-1} \right) dx \\ &= \sum_{l=1}^k \left( \log \left( \sum_{m=1}^l a_m + \sum_{m=1}^{l-1} b_m + 1 \right) - \log \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + 1 \right) \right. \\ &\quad \left. - \log \left( \sum_{m=1}^l a_m + \sum_{m=1}^{l-1} b_m + 1 \right) + \log \left( \sum_{m=2}^l a_m + \sum_{m=1}^{l-1} b_m + 1 \right) \right) \end{aligned}$$

$$= \sum_{l=1}^k \left( \log (|[A_1, A_l]| + 1) - \log (|[B_1, A_l]| + 1) - \log (|[A_1, B_l]| + 1) + \log (|[B_1, B_l]| + 1) \right),$$

using the definition e.g.  $|[A_1, B_l]| = \sum_{j=1}^l (a_j + b_j)$ .  $\blacktriangleleft$

Next, we are going to sum up the lower bounds in Lemma 14 to obtain a lower bound on  $f(A, B)$ . To write the sum nicely, we use a simple observation on the signs of the logarithm in our sum. Let us call a contiguous block  $[S, T]$  odd if there are an odd number of sets in  $\{A_i\}_{i=1}^k \cup \{B_i\}_{i=1}^k$ , and even otherwise. Note that the odd blocks are exactly those with the first and last sets from the same partition  $A$  or  $B$ , e.g.  $[A_i, A_j], [B_i, B_j]$ . With this definition, the lower bound on  $f(A, B)$  can be written as follows.

► **Lemma 16.** *Using the definitions and notations in this subsection,*

$$\sqrt{2C} \cdot f(A, B) \geq \sum_{S \neq T: [S, T] \text{ odd}} \log (|[S, T]| + 1) - \sum_{S \neq T: [S, T] \text{ even}} \log (|[S, T]| + 1) - (k-1) \log(n+1),$$

where the sum is over  $S, T \in \{A_i\}_{i=1}^k \cup \{B_i\}_{i=1}^k$ .

**Proof.** We sum the inequalities in Lemma 14 from  $1 \leq i \leq k$ . On the right hand side of the inequality in Lemma 14, we see that all contiguous blocks starting from  $A_i$  or  $B_i$  are in the sum, with the odd blocks positive and even blocks negative. Thus, summing over all  $A_i$ , every contiguous block is counted once as it is uniquely determined by the starting and ending sets, except for the whole cycle which appears once on the right hand side for every  $i$  with a negative sign.  $\blacktriangleleft$

To prove a lower bound on the right hand side of Lemma 16, the idea is to use the following concavity property.

► **Lemma 17.** *For  $k \geq 2$ , consider the function*

$$h : (a_1, b_1, \dots, a_k, b_k) \rightarrow \sum_{S \neq T: [S, T] \text{ odd}} \log (|[S, T]| + 1) - \sum_{S \neq T: [S, T] \text{ even}} \log (|[S, T]| + 1) - (k-1) \log(n+1),$$

where the sum is over  $S, T \in \{A_i\}_{i=1}^k \cup \{B_i\}_{i=1}^k$  and so  $|[S, T]|$  depends on  $a_1, b_1, \dots, a_k, b_k$ . Then, for all positive  $j$ , the function

$$g : x \rightarrow h(x, b_1, s - x, b_2, \dots, a_k, b_k),$$

obtained by fixing non-negative integers  $b_1, b_2, a_3, b_3, \dots, a_k, b_k$  as the size of the other sets and  $s$  as the sum of  $a_1 + a_2$ , is concave on  $x \in [0, s]$ .

**Proof.** To prove concavity, we use the second derivative test, where  $g$  is concave if the second derivative  $g''$  is non-positive. We write  $g$  as  $g_0 + g_1(x) + g_2(x)$ , where the  $g_1(x)$  consists of all the log terms which contain  $A_1$  but not  $A_2$ , and similarly  $g_2(x)$  consists of all the log terms which contain only  $A_2$  but not  $A_1$ . The remaining terms are in  $g_0$ , which either contain both  $A_1$  and  $A_2$  or none of  $A_1$  and  $A_2$ . Note that these terms are independent of  $x$ , because if a block  $[S, T]$  contains both  $A_1$  and  $A_2$  then its size  $|[S, T]|$  is the same even when we change  $x$ , so these terms can be ignored when we compute derivatives.

### 36:16 On the Houdré-Tetali Conjecture About an Isoperimetric Constant of Graphs

Let us focus on  $g_1(x)$  first. The blocks that contain  $A_1$  but not  $A_2$  must be of the form  $[S, A_1]$  or  $[S, B_1]$  for some set  $S$ . Let  $\sigma([S, T])$  denote the parity of the block  $[S, T]$ . Note that the parity of  $[S, A_1]$  and  $[S, B_1]$  are different, and so

$$\begin{aligned}
 g_1''(x) &= \sum_S (-1)^{\sigma([S, A_1])+1} \left( (\log(|[S, A_1]| + 1))'' - (\log(|[S, B_1]| + 1))'' \right) \\
 &= \sum_S (-1)^{\sigma([S, A_1])+1} \left( \log(|[S, B_k]| + x + 1)'' - (\log(|[S, B_k]| + x + b_1 + 1))'' \right) \\
 &= \sum_S (-1)^{\sigma([S, A_1])} \left( (|[S, B_k]| + x + 1)^{-2} - (|[S, B_k]| + x + b_1 + 1)^{-2} \right) \\
 &= \sum_S (-1)^{\sigma([S, A_1])} \left( b_1 \cdot (|[S, B_k]| + x + 1)^{-2} \cdot (|[S, B_k]| + x + b_1 + 1)^{-1} \right. \\
 &\quad \left. + b_1 \cdot (|[S, B_k]| + x + 1)^{-1} \cdot (|[S, B_k]| + x + b_1 + 1)^{-2} \right),
 \end{aligned}$$

where the sum is over  $S \in \{A_i\}_{i=1}^k \cup \{B_i\}_{i=1}^k$ . In the special case when  $S = A_1$ , we violate our own notation and let  $|[A_1, B_k]| = 0$  in this proof; all other cases are still the same.

When  $b_1 = 0$ , the sum equals zero and we are done, so assume  $b_1 \neq 0$ . To see that  $g_1''(x)$  is negative, we pair up the terms with  $S = B_i$  and  $S = A_{i+1}$  with indices taken modulo  $k$  so that

$$\begin{aligned}
 \frac{1}{b_1} \cdot g_1''(x) &= \sum_{i=1}^k \left[ (|[B_i, B_k]| + x + 1)^{-2} \cdot (|[B_i, B_k]| + x + b_1 + 1)^{-1} \right. \\
 &\quad \left. + (|[B_i, B_k]| + x + 1)^{-1} \cdot (|[B_i, B_k]| + x + b_1 + 1)^{-2} \right. \\
 &\quad \left. - (|[A_{i+1}, B_k]| + x + 1)^{-2} \cdot (|[A_{i+1}, B_k]| + x + b_1 + 1)^{-1} \right. \\
 &\quad \left. - (|[A_{i+1}, B_k]| + x + 1)^{-1} \cdot (|[A_{i+1}, B_k]| + x + b_1 + 1)^{-2} \right] \\
 &< 0,
 \end{aligned}$$

where the inequality holds because  $|[A_{i+1}, B_k]| < |[B_i, B_k]|$  and so each summand is negative (recall the special case that  $|[A_1, B_k]| = 0$  in this proof).

The function  $g_2(x)$  is handled analogously in view of the symmetry of the second derivative of the logarithm. This proves that  $g$  is concave.  $\blacktriangleleft$

With the concavity property, we can apply a simple “swapping/merging” argument to reduce to the case when there is only one contiguous set, i.e.  $k = 1$ , and then finish the proof.

By concavity, the function  $g(x)$  attains its minimum at one of the endpoints, and so

$$h(a_1, \dots, b_n) \geq \min \{h(0, b_1, a_1 + a_2, b_2, \dots, a_n, b_n), h(a_1 + a_2, b_1, 0, b_2, \dots, a_n, b_n)\}$$

The next observation is that when one set has size zero, we can merge the two adjacent sets in the same partition into one. More formally, let  $b_1 = 0$  without loss of generality, we claim that

$$h(a_1, 0, a_2, b_2, \dots, a_k, b_k) = h(a_1 + a_2, b_2, \dots, a_k, b_k).$$

To see this, note that  $|[S, A_1]|$  and  $|[S, B_1]|$  have the same values but they have different signs so the terms involving them cancel out each other, and similarly the terms involving  $|[B_1, S]|$  and  $|[A_2, S]|$  cancel out each other. Therefore, in  $h(a_1, 0, a_2, b_2, \dots, a_k, b_k)$ , there are no terms ending with  $A_1$  or  $B_1$  and no terms beginning with  $B_1$  or  $A_2$ , and all the remaining terms have a one-to-one correspondence with the terms in  $h(a_1 + a_2, b_2, \dots, a_k, b_k)$ .



This reduces  $k$  by one. Repeating the same argument until  $k = 1$ , we see that

$$\sqrt{2C} \cdot f(A, B) \geq h(|A|, n - |A|) = \log(|A| + 1) + \log(n - |A| + 1) - \log(n + 1),$$

and thus

$$\begin{aligned} \varphi_{\frac{1}{2}}(G) &= \min_{A:|A| \leq \frac{n}{2}} \frac{f(A, B)}{|A|} \gtrsim \min_{l: l \leq \frac{n}{2}} \frac{h(l, n - l)}{l} \\ &= \min_{l: l \leq \frac{n}{2}} \frac{\log(l + 1) + \log(n - l + 1) - \log(n + 1)}{l}, \end{aligned}$$

where we used that  $C$  is upper bounded by an absolute constant.

It remains to lower bound the right hand side. Since  $l \leq \frac{n}{2}$ , it follows that  $\log(n - l + 1) - \log(n + 1) \geq \log((n + 1)/2) - \log(n + 1) = -\log 2$ , and so

$$\frac{\log(k + 1) + \log(n - k + 1) - \log(n + 1)}{k} \geq \frac{\log \frac{k+1}{2}}{k} \gtrsim \frac{\log \frac{k+1}{2}}{\frac{k+1}{2}} \geq \frac{\log n}{n},$$

where the last inequality is because  $\frac{\log n}{n}$  is decreasing for  $n \geq 3$  and for  $1 \leq k \leq 4$  the last inequality clearly holds when  $n$  is large enough. This concludes the proof of Theorem 4.

## Concluding Remarks and Open Questions

We believe that the same analysis of  $\varphi_p(G)$  can be extended to other generalizations of Cheeger's inequality in [15, 11], and also to the directed edge conductance using the recent notions of reweighted eigenvalues in [19, 12, 13, 14]. We leave it as an open question to find a counterexample where the transition matrix is the simple random walk matrix of a graph.

---

## References

- 1 Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6:83–96, 1986.
- 2 Noga Alon and Vitali Milman.  $\mathfrak{h}_1$ , isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.
- 3 Deeparnab Chakrabarty and C. Seshadhri. An  $o(n)$  monotonicity tester for boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2), 2016.
- 4 Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Problems in Analysis*, pages 195–199. Princeton University Press, 1970.
- 5 Fan Chung. Laplacians and cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.
- 6 Ronen Eldan and Renan Gross. Concentration on the boolean hypercube via pathwise stochastic analysis. *Inventiones Mathematicae*, 230:935–994, 2022.
- 7 Ronen Eldan, Guy Kindler, Noam Lifshitz, and Dor Minzer. Isoperimetric inequalities made simpler. Technical report, ArXiv preprint, 2204.06686, 2022.
- 8 Christian Houdré and Prasad Tetali. Isoperimetric invariants for product Markov chains and graph products. *Combinatorica*, 24:359–388, 2004.
- 9 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorem. *SIAM Journal on Computing*, 47(6), 2018.
- 10 Tsz Chiu Kwok, Lap Chi Lau, and Yin Tat Lee. Improved cheeger's inequality and analysis of local graph partitioning using vertex expansion and expansion profile. *SIAM Journal on Computing*, 46(3):890–910, 2017.
- 11 Tsz Chiu Kwok, Lap Chi Lau, Yin Tat Lee, Shayan Oveis Gharan, and Luca Trevisan. Improved cheeger's inequality: Analysis of spectral partitioning algorithms through higher order spectral gap. In *Proceedings of the 45th Annual Symposium on Theory of Computing (STOC)*, pages 11–20, 2013.

- 12 Tsz Chiu Kwok, Lap Chi Lau, and Kam Chuen Tung. Cheeger inequalities for vertex expansion and reweighted eigenvalues. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 366–377, 2022.
- 13 Lap Chi Lau, Kam Chuen Tung, and Robert Wang. Cheeger inequalities for directed graphs and hypergraphs using reweighted eigenvalues. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1834–1847, 2023.
- 14 Lap Chi Lau, Kam Chuen Tung, and Robert Wang. Fast algorithms for directed graph partitioning using flows and reweighted eigenvalues. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 591–624, 2024.
- 15 James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multi-way spectral partitioning and higher-order cheeger inequalities. In *Proceedings of the 44th Annual Symposium on Theory of Computing (STOC)*, pages 1117–1130, 2012.
- 16 David Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Society, 2017.
- 17 Grigory Margulis. Probabilistic characteristics of graphs with large connectivity. *Problemy Peredachi Informatsii*, 10(2):101–108, 1977.
- 18 Ben Morris and Yuval Peres. Evolving sets, mixing and heat kernel bounds. *Probability Theory and Related Fields*, 133:245–266, 2005.
- 19 Sam Olesker-Taylor and Luca Zanetti. Geometric bounds on the fastest mixing Markov chain. In *the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, 2022.
- 20 Michel Talagrand. Isoperimetry, logarithmic sobolev inequalities on the discrete cube, and margulis’ graph connectivity theorem. *Geometric and Functional Analysis*, 3(3):295–314, 1993.
- 21 Yuichi Yoshida. Nonlinear laplacian for digraphs and its applications to network analysis. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 483–492, 2016.

# Nearly Optimal Bounds for Sample-Based Testing and Learning of $k$ -Monotone Functions

Hadley Black   

University of California, San Diego, USA

---

## Abstract

We study monotonicity testing of functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  using *sample-based* algorithms, which are only allowed to observe the value of  $f$  on points drawn independently from the uniform distribution. A classic result by Bshouty-Tamon (J. ACM 1996) proved that monotone functions can be learned with  $\exp(\tilde{O}(\min\{\frac{1}{\varepsilon}\sqrt{d}, d\}))$  samples and it is not hard to show that this bound extends to testing. Prior to our work the only lower bound for this problem was  $\Omega(\sqrt{\exp(d)/\varepsilon})$  in the small  $\varepsilon$  parameter regime, when  $\varepsilon = O(d^{-3/2})$ , due to Goldreich-Goldwasser-Lehman-Ron-Samorodnitsky (Combinatorica 2000). Thus, the sample complexity of monotonicity testing was wide open for  $\varepsilon \gg d^{-3/2}$ . We resolve this question, obtaining a nearly tight lower bound of  $\exp(\Omega(\min\{\frac{1}{\varepsilon}\sqrt{d}, d\}))$  for all  $\varepsilon$  at most a sufficiently small constant. In fact, we prove a much more general result, showing that the sample complexity of  $k$ -monotonicity testing and learning for functions  $f: \{0, 1\}^d \rightarrow [r]$  is  $\exp(\Omega(\min\{\frac{rk}{\varepsilon}\sqrt{d}, d\}))$ . For testing with one-sided error we show that the sample complexity is  $\exp(\Omega(d))$ .

Beyond the hypercube, we prove nearly tight bounds (up to polylog factors of  $d, k, r, 1/\varepsilon$  in the exponent) of  $\exp(\tilde{\Theta}(\min\{\frac{rk}{\varepsilon}\sqrt{d}, d\}))$  on the sample complexity of testing and learning measurable  $k$ -monotone functions  $f: \mathbb{R}^d \rightarrow [r]$  under product distributions. Our upper bound improves upon the previous bound of  $\exp(\tilde{O}(\min\{\frac{k}{\varepsilon^2}\sqrt{d}, d\}))$  by Harms-Yoshida (ICALP 2022) for Boolean functions ( $r = 2$ ).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Mathematics of computing  $\rightarrow$  Probabilistic algorithms

**Keywords and phrases** Property testing, learning, Boolean functions, monotonicity,  $k$ -monotonicity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.37

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2310.12375>

**Funding** *Hadley Black:* This work was done while the author was at UCLA supported by NSF award AF:Small 2007682, NSF Award: Collaborative Research Encore 2217033.

**Acknowledgements** We would like to thank Eric Blais and Nathaniel Harms for helpful discussions during the early stages of this work and for their thoughtful feedback. We would also like to thank the anonymous reviewers whose comments helped significantly to improve this write up.

## 1 Introduction

A function  $f: \mathcal{X} \rightarrow \mathbb{R}$  over a partial order  $\mathcal{P} = (\mathcal{X}, \preceq)$  is  $k$ -monotone if there does not exist a chain of  $k + 1$  points  $x_1 \prec x_2 \prec \dots \prec x_{k+1}$  for which (a)  $f(x_{i+1}) - f(x_i) < 0$  when  $i$  is odd and (b)  $f(x_{i+1}) - f(x_i) > 0$  when  $i$  is even. When  $k = 1$ , these are the *monotone* functions, which are the non-decreasing functions with respect to  $\preceq$ . Monotone and  $k$ -monotone *Boolean* functions over domains  $\{0, 1\}^d$ ,  $[n]^d$ , and  $\mathbb{R}^d$  have been the focus of a significant amount of research in property testing and computational learning theory. We give an overview of the literature in Section 1.4.



© Hadley Black;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 37; pp. 37:1–37:23



Leibniz International Proceedings in Informatics

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

The field of property testing is concerned with the design and analysis of sub-linear time randomized algorithms for determining if a function has, or is far from having, some specific property. A key aspect in the definition of a property testing algorithm is the type of access it has to the function. Early works on property testing, e.g. [61, 44], focused on the notion of *query-based* testers, which are allowed to observe the value of the function on any point of their choosing, and since then this has become the standard model. The weaker notion of *sample-based* testers, which can only view the function on independent uniform samples, was also considered by [44] and has received some attention over the years, see e.g. [51, 3, 41, 45, 37, 38]. Sample-based algorithms are considered more natural in many settings, for example in computational learning theory, where they are the standard model. In fact, sample-based testing and learning are closely related problems; given a learning algorithm, it is always possible to design a testing algorithm with the same sample complexity, up to an additive  $\text{poly}(1/\varepsilon)$  factor<sup>1</sup>.

For many fundamental properties, there is still a large gap between how much we know in the query-based vs the sample-based models. Monotonicity (and  $k$ -monotonicity) is such a property; despite a vast body of research on query-based monotonicity testing over the hypercube  $\{0, 1\}^d$ , the only work we know of which considers this problem in the sample-based model is [43], who gave an upper bound of  $O(\sqrt{2^d/\varepsilon})$  and a matching lower bound for the case when  $\varepsilon = O(d^{-3/2})$  on the number of samples needed to test monotonicity of functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$ . The upper bound for learning monotone Boolean functions due to [23, 54] also implies a testing upper bound of  $\exp(\tilde{O}(\frac{1}{\varepsilon}\sqrt{d}))$ . Thus, this question has been wide open for  $\varepsilon \gg d^{-3/2}$ .

Our work addresses this gap in the monotonicity testing literature, proving a lower bound which matches the learning upper bound for all  $\varepsilon$  at most some constant, up to a factor of  $\log d$  in the exponent. More generally, we prove tight lower bounds for  $k$ -monotonicity testing of functions,  $f: \{0, 1\}^d \rightarrow [r]$ , i.e. functions with image size at most  $r$ . To round out our results, we also give an improved learning algorithm for  $k$ -monotone functions over  $\mathbb{R}^d$  under product distributions whose sample complexity matches our sample-based testing lower bound, up to poly-logarithmic factors in the exponent.

## 1.1 Results

Before explaining our results and the context for them, we first provide some terminology and basic notation. Given a domain  $\mathcal{X}$  and a distribution  $\mu$  over  $\mathcal{X}$ , we denote the Hamming distance between two functions  $f, g: \mathcal{X} \rightarrow \mathbb{R}$  under  $\mu$  by  $d_\mu(f, g) = \mathbb{P}_{x \sim \mu}[f(x) \neq g(x)]$ . We say that  $f$  is  $\varepsilon$ -far from  $k$ -monotone if  $d_\mu(f, g) \geq \varepsilon$  for every  $k$ -monotone function  $g$ . The results in this paper pertain to sample-based testing and learning of  $k$ -monotone functions with respect to Hamming distance. We use the following terminology:

- The *example oracle* for  $f$  under  $\mu$ , denoted by  $EX(f, \mu)$ , when queried, generates an example  $(x, f(x))$  where  $x$  is sampled according to  $\mu$ .
- A *sample-based  $k$ -monotonicity tester* under  $\mu$  is a randomized algorithm which is given access to  $EX(f, \mu)$  for an arbitrary input function  $f$  and satisfies the following: (a) if  $f$  is  $k$ -monotone, then the algorithm accepts with probability at least  $2/3$ , and (b) if  $f$  is  $\varepsilon$ -far from  $k$ -monotone, then the algorithm rejects with probability at least  $2/3$ . The tester has *one-sided error* if in case (a) it accepts with probability 1.

<sup>1</sup> See Lemma C.1 in the full version of the paper for a precise statement. Also, note that if the learning algorithm is *proper*, then the time complexity is also preserved. If the learning algorithm is *improper*, then there is a time complexity blow-up, but the sample complexity is still preserved.

- A *sample-based learning algorithm for  $k$ -monotone functions* under  $\mu$  is a randomized algorithm which is given access to  $EX(f, \mu)$  for an arbitrary  $k$ -monotone input function  $f$  and outputs a hypothesis  $h$  such that  $d_\mu(f, h) \leq \varepsilon$  with probability at least  $1 - \delta$ . If left unspecified,  $\delta = 1/3$ .

In all of the above definitions if  $\mu$  is unspecified, then it is the uniform distribution. Testing and learning are closely related problems; any sample-based learning algorithm can be used to construct a sample-based tester with the same sample complexity. We refer to this transformation as the testing-by-learning reduction and although this is not a new idea we provide a proof in Section C in the full version of the paper for completeness.

Finally, we recall some important learning theory terminology. A learning algorithm for concept class  $\mathcal{C}$  is called *proper* if it always outputs a hypothesis  $h \in \mathcal{C}$ , and is called *improper* if it is allowed to output arbitrary  $h$ . Given a function  $f$ , and a concept class  $\mathcal{C}$ , let  $d(f, \mathcal{C}) = \min_{g \in \mathcal{C}} d(f, g)$ . An *agnostic proper* learner is one which, given *any*  $f$  (not necessarily in  $\mathcal{C}$ ), outputs a hypothesis  $h \in \mathcal{C}$  for which  $d(f, h) \leq d(f, \mathcal{C}) + \varepsilon$  with probability at least  $1 - \delta$ .

### 1.1.1 Sample-Based Testing and Learning on the Hypercube

The problem of *learning* monotone Boolean functions over the hypercube  $\{0, 1\}^d$  was studied by [23] who proved an upper bound<sup>2</sup> of  $\exp(O(\min\{\frac{1}{\varepsilon}\sqrt{d} \log d, d\}))$  for improper learning and very recently by [54, 55] who obtained the same upper bound for agnostic proper learning. The improper learning upper bound was extended by [17] who showed an upper bound of  $\exp(O(\min\{\frac{k}{\varepsilon}\sqrt{d} \log d, d\}))$  and a nearly matching lower bound of  $\exp(\Omega(\min\{\frac{k}{\varepsilon}\sqrt{d}, d\}))$  for learning  $k$ -monotone Boolean functions for any  $k \geq 1$ . The testing-by-learning reduction shows that their upper bound also holds for sample-based *testing*. The only prior lower bound for sample-based testing that we're aware of is  $\Omega(\sqrt{2^d/\varepsilon})$  when  $\varepsilon = O(d^{-3/2})$  and  $k = 1$  [43, Theorem 5]. Our main result is the following much more general lower bound for this problem, which we prove in Section 3.

► **Theorem 1 (Testing Lower Bound).** *There is an absolute constant  $c > 0$  such that for all  $\varepsilon \leq c$ , every sample-based  $k$ -monotonicity tester for functions  $f: \{0, 1\}^d \rightarrow [r]$  under the uniform distribution has sample complexity*

$$\exp\left(\Omega\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}, d\right\}\right)\right).$$

Even for the special case of sample-based monotonicity testing of Boolean functions ( $k = 1$  and  $r = 2$ ), Theorem 1 is already a new result, which matches the upper bound for learning by [23] and is the first lower bound to hold for  $\varepsilon \gg d^{-3/2}$ . Moreover, our lower bound is much more general, holding for all  $r, k$ , and is optimal in all parameters,  $d, r, k, \varepsilon$ , up to a  $\log d$  factor in the exponent. We show a nearly matching upper bound in Theorem 3.

We also note that the testing-by-learning reduction implies that the same lower bound holds for *learning* with samples. As we mentioned, this result was already known for Boolean

<sup>2</sup> We remark that any function over  $\{0, 1\}^d$  can be learned exactly with  $O(d2^d) = \exp(O(d))$  samples by a coupon-collector argument. Combining this with the  $\exp(O(\frac{1}{\varepsilon}\sqrt{d} \log d))$  upper bound by [23] yields  $\exp(O(\min\{\frac{1}{\varepsilon}\sqrt{d} \log d, d\}))$ . We use this slightly clunkier notation involving the min to emphasize that our upper and lower bounds are nearly matching in all parameter regimes.

## 37:4 Sample-Based Testing and Learning of $k$ -Monotone Functions

functions (the  $r = 2$  case) [17], but the general case of  $r \geq 2$  was not known prior to our work<sup>3</sup>.

► **Corollary 2** (Learning Lower Bound). *There is an absolute constant  $c > 0$  such that for every  $\varepsilon \leq c$ , every sample-based uniform-distribution learning algorithm for  $k$ -monotone functions  $f: \{0, 1\}^d \rightarrow [r]$  has sample complexity*

$$\exp\left(\Omega\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}, d\right\}\right)\right).$$

On the upper bound side, a relatively straightforward argument extends the learning algorithm of [17] for Boolean  $k$ -monotone functions, to  $k$ -monotone functions with image size at most  $r$ . We give a short proof in Section 1.5. This shows that our lower bounds in Theorem 1 and Corollary 2 are tight up to a factor of  $\log d$  in the exponent.

► **Theorem 3** (Learning Upper Bound for Hypercubes). *There is a uniform-distribution learning algorithm for  $k$ -monotone functions  $f: \{0, 1\}^d \rightarrow [r]$  which achieves error at most  $\varepsilon$  with time and sample complexity*

$$\exp\left(O\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}\log d, d\right\}\right)\right).$$

The testing-by-learning reduction again gives us the following corollary.

► **Corollary 4** (Testing Upper Bound for Hypercubes). *There is a sample-based  $k$ -monotonicity tester for functions  $f: \{0, 1\}^d \rightarrow [r]$  with sample complexity*

$$\exp\left(O\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}\log d, d\right\}\right)\right).$$

Lastly, we consider the problem of sample-based testing with *one-sided error*. For monotonicity testing of functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  with *non-adaptive queries*, we know that one-sided and two-sided error testers achieve the same query-complexity (up to  $\text{polylog}(d, 1/\varepsilon)$  factors): there is a  $\tilde{O}(\sqrt{d}/\varepsilon^2)$  one-sided error upper bound due to [53] and a  $\tilde{\Omega}(\sqrt{d})$  two-sided error lower bound due to [33]. We show that the situation is quite different for *sample-based* monotonicity testing; while the sample complexity of two-sided error testers is  $\exp(\Theta(\min\{\frac{1}{\varepsilon}\sqrt{d}, d\}))$ , one-sided error testers require  $\exp(\Theta(d))$  samples for all  $\varepsilon$ .

► **Theorem 5** (Testing with One-Sided Error). *For every  $d, r, k$ , and  $\varepsilon > 0$ , sample-based  $k$ -monotonicity testing of functions  $f: \{0, 1\}^d \rightarrow [r]$  with one-sided error requires  $\exp(\Theta(d))$  samples.*

### 1.1.2 Sample-Based Testing and Learning in Continuous Product Spaces

Learning  $k$ -monotone Boolean-valued functions has also been studied over  $\mathbb{R}^d$  with respect to product measures by [49] who gave an upper bound of  $\exp(\tilde{O}(\min\{\frac{k}{\varepsilon^2}\sqrt{d}, d\}))$  where  $\tilde{O}(\cdot)$  hides  $\text{polylog}$  factors of  $d, k$ , and  $1/\varepsilon$ . Our next result gives an upper bound which improves the dependence on  $\varepsilon$  from  $1/\varepsilon^2$  to  $1/\varepsilon$  in the exponent. By the same approach we used to generalize the upper bound in Theorem 3 to arbitrary  $r \geq 2$ , we get the same generalization for product spaces. We obtain the following upper bound which matches our lower bound for  $\{0, 1\}^d$  in Theorem 1 up to  $\text{polylog}$  factors of  $d, k, r$ , and  $1/\varepsilon$ . We say that a function  $f: \mathbb{R}^d \rightarrow [r]$  is *measurable* if the set  $f^{-1}(i)$  is measurable for every  $i \in [r]$ .

<sup>3</sup> It is possible that the techniques from [17] could be extended to provide an alternative proof of Corollary 2, but we have not checked whether this is the case.

► **Theorem 6** (Learning Upper Bound for Product Spaces). *Given an arbitrary product measure  $\mu$ , there is a learning algorithm under  $\mu$  for measurable  $k$ -monotone functions  $f: \mathbb{R}^d \rightarrow [r]$  with time and sample complexity*

$$\exp\left(\tilde{O}\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}, d\right\}\right)\right).$$

The  $\tilde{O}(\cdot)$  hides polylogarithmic dependencies on  $d, r, k$ , and  $1/\varepsilon$ .

We prove Theorem 6 in Section 4. Once again the testing-by-learning reduction gives us the following corollary for sample-based testing.

► **Corollary 7** (Testing Upper Bound for Product Spaces). *Given an arbitrary product measure  $\mu$ , there is a  $k$ -monotonicity tester for measurable functions  $f: \mathbb{R}^d \rightarrow [r]$  under  $\mu$  with sample complexity*

$$\exp\left(\tilde{O}\left(\min\left\{\frac{rk}{\varepsilon}\sqrt{d}, d\right\}\right)\right).$$

The  $\tilde{O}(\cdot)$  hides polylogarithmic dependencies on  $d, r, k$ , and  $1/\varepsilon$ .

## 1.2 Proof Overviews

In this section we give an overview of our proofs for Theorem 1 and Theorem 6.

### 1.2.1 The Testing Lower Bound for Hypercubes

Our proof of Theorem 1 uses a family functions known as *Talagrand's random DNFs* introduced by [63] which have been used by [4] and [33] to prove lower bounds for monotonicity testing of Boolean functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  against adaptive and non-adaptive query-based testers. Very recently, they have also been used to prove lower bounds for tolerant monotonicity testing [29] and for testing convexity of sets in  $\{-1, 0, 1\}^d$  [10].

To understand our construction, let us first consider the special case of monotonicity of Boolean functions, i.e.  $k = 1$  and  $r = 2$ . We think of a DNF term as a point  $t \in \{0, 1\}^d$  which is said to be satisfied by  $x \in \{0, 1\}^d$  if  $t \preceq x$ , where  $\preceq$  denotes the standard bit-wise partial order over  $\{0, 1\}^d$ . The *width* of a term  $t$  is its Hamming weight,  $|t|$ , and the width of a DNF is the max width among its terms. Consider  $N$  randomly chosen terms  $t^1, \dots, t^N$  each of width  $|t^j| = w$ . We will see later how to choose  $N$  and  $w$ . Let  $B := \{x: \frac{d}{2} \leq |x| \leq \frac{d}{2} + \varepsilon\sqrt{d}\}$  and for each  $j \in [N]$ , let

$$U_j := \{x \in B: t^j \preceq x \text{ and } t^{j'} \not\preceq x \text{ for all } j' \neq j\}$$

be the set of points in  $B$  which satisfy  $t^j$  and no other terms. Let  $U := \bigcup_{j \in [N]} U_j$ . Now observe that any two points lying in different  $U_j$ 's are *incomparable* and therefore independently embedding an arbitrary monotone function into each  $U_j$  will result in a function which globally is monotone if one defines the function outside of  $U$  appropriately. Using this fact we can define two distributions  $\mathcal{D}_{\text{yes}}$  and  $\mathcal{D}_{\text{no}}$  as follows. Let  $A$  denote the set of points in  $x \in \{0, 1\}^d$  for which either  $|x| > \frac{d}{2} + \varepsilon\sqrt{d}$  or  $x \in B$  and  $t^j, t^{j'} \preceq x$  for two different terms  $j \neq j'$ .

- $f \sim \mathcal{D}_{\text{yes}}$  is drawn by setting  $f(x) = 1$  if and only if  $x \in A \cup \left(\bigcup_{j \in T} U_j\right)$  where  $T \subseteq [N]$  contains each  $j \in [N]$  with probability  $1/2$ , independently. Such a function is always monotone.

- $f \sim \mathcal{D}_{\text{no}}$  is drawn by setting  $f(x) = 1$  if and only if  $x \in A \cup R$  where  $R$  contains each  $x \in U$  with probability  $1/2$ , independently. Such a function will be  $\Omega(|U| \cdot 2^{-d})$ -far from monotone with probability  $\Omega(1)$  since its restriction with  $U$  is uniformly random.

Now, each  $x \in U$  satisfies  $\mathbb{P}_{f \sim \mathcal{D}_{\text{yes}}}[f(x) = 1] = \mathbb{P}_{f \sim \mathcal{D}_{\text{no}}}[f(x) = 1] = 1/2$  and for both distributions the events  $f(x) = 1$  and  $f(y) = 1$  are independent when  $x, y$  lie in different  $U_j$ 's. Therefore, any tester will need to see at least two points from the same  $U_j$  to distinguish  $\mathcal{D}_{\text{yes}}$  and  $\mathcal{D}_{\text{no}}$ . Roughly speaking, by birthday paradox this gives a  $\Omega(\sqrt{N})$  lower bound on the number of samples. The lower bound is thus determined by the maximum number of terms  $N$  that can be used in the construction for which  $|U| = \Omega(\varepsilon 2^d)$ .

So how are  $N$  and  $w$  chosen? By standard concentration bounds, we have  $|B| = \Omega(\varepsilon 2^d)$  and observe that a point  $x \in B$  satisfies a random term with probability exactly  $(|x|/d)^w$ . We need  $U$  to contain a *constant fraction* of  $B$ , i.e. we need  $x$  to satisfy exactly 1 term with constant probability. The expected number of satisfied terms is  $N \cdot (|x|/d)^w$  and, roughly speaking, we need this value to be  $\Theta(1)$  for all  $x \in B$ . Applying this constraint to the case when  $|x| = d/2$  forces us to pick  $N \approx 2^w$ . Now when  $|x| = d/2 + \varepsilon\sqrt{d}$ , the expected number of satisfied terms is  $N \cdot 2^{-w} \cdot (1 + 2\varepsilon/\sqrt{d})^w \approx (1 + 2\varepsilon/\sqrt{d})^w$  and we are forced to choose  $w \approx \sqrt{d}/\varepsilon$ . The lower bound for sample-based monotonicity testing of  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  is then  $\Omega(\sqrt{N}) \approx \exp(\Omega(\sqrt{d}/\varepsilon))$ .

Let us now think about generalizing this construction to testing  $k$ -monotonicity of functions  $f: \{0, 1\}^d \rightarrow [r]$ . The moral of the above argument is that the permitted number of terms is controlled by the number of distinct Hamming weights in the set  $B$ . We observe that for larger values of  $k$  and  $r$  we can partition  $B$  into  $k(r-1)$  blocks as  $B := B_1 \cup B_2 \cup \dots \cup B_{k(r-1)}$  each with a window of Hamming weights of size only  $\frac{\varepsilon\sqrt{d}}{k(r-1)}$ . We are able to essentially repeat the above construction independently within each block wherein we can set  $w \approx \frac{k(r-1)\sqrt{d}}{\varepsilon}$  and consequently  $N \approx 2^{\frac{k(r-1)\sqrt{d}}{\varepsilon}}$ .

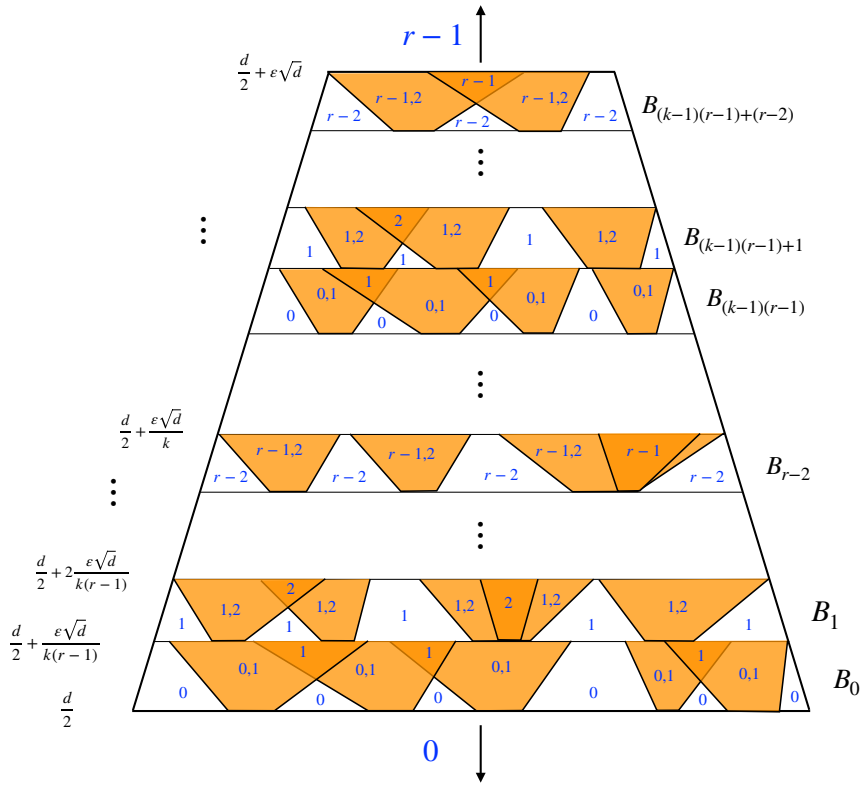
For each block  $i \in [k(r-1)]$ , the random Talagrand DNF within block  $B_i$  is defined analogously to the above construction, except that it assigns function values from  $\{i \bmod (r-1), i \bmod (r-1) + 1\}$ , instead of  $\{0, 1\}$ . See Figure 1 for an illustration. Since there are  $k(r-1)$  blocks in total, the distribution  $\mathcal{D}_{\text{yes}}$  only produces  $k$ -monotone functions. At the same time, a function  $f \sim \mathcal{D}_{\text{no}}$  assigns uniform random  $\{a, a+1\}$  values within each block  $B_{m(r-1)+a}$ . This results in a large number of long chains through  $B_a \cup B_{(r-1)+a} \cup \dots \cup B_{(k-1)(r-1)+a}$  which alternate between function value  $a$  and  $a+1$ . Considering the union of all such chains for  $a = 0, 1, \dots, r-2$  shows that  $f$  is  $\Omega(\varepsilon)$ -far from  $k$ -monotone with probability  $\Omega(1)$ .

## 1.2.2 The Learning Upper Bound for Product Spaces

As we discussed in Section 1.1, it suffices to prove Theorem 6 for the case of  $r = 2$ , i.e. learning functions  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  under a product measure  $\mu$ . We use a downsampling technique to reduce this problem to learning a discretized proxy of  $f$  over a hypergrid  $[N]^d$  where  $N = \Theta(kd/\varepsilon)$  with mild label noise. This technique has been used in previous works [46, 12, 49] and our proof borrows many technical details from [49].

Next, for  $N$  which is a power of 2, we observe that a  $k$ -monotone function  $f: [N]^d \rightarrow \{\pm 1\}$  can be viewed as a  $k$ -monotone function over the hypercube  $\{\pm 1\}^{d \log N}$  by mapping each point  $x \in [N]^d$  to its bit-representation. We can then leverage a result of [17] which shows that all but a  $\varepsilon$ -fraction of the mass of the Fourier coefficients of  $k$ -monotone Boolean functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  is concentrated on the terms with degree at most  $\frac{k\sqrt{d}}{\varepsilon}$ . We can then use the Low-Degree Algorithm introduced by [57] which was shown to work under random classification noise by [50].





■ **Figure 1** An illustration of the construction used in our proof of Theorem 1. The image represents the set of points in the hypercube  $\{0, 1\}^d$  with Hamming weight in the interval  $[\frac{d}{2}, \frac{d}{2} + \epsilon\sqrt{d}]$ , increasing from bottom to top. The numbers on the left denote the Hamming weight of the points lying in the adjacent horizontal line. The  $B_i$  blocks are the sets of points contained between two adjacent horizontal lines. Each orange shaded region within  $B_i$  represents the set of points satisfied by a term  $t^{i,j}$ . The blue numbers represent the value that functions in the support of  $\mathcal{D}_{\text{yes}}$  and  $\mathcal{D}_{\text{no}}$  can take. We have used the notation “ $r - 1, 2$ ” as shorthand for  $r - 2, r - 1$ .

### 1.3 Discussion and Open Questions

Our results for sample-based testing and learning over the hypercube are tight up to a  $\log d$  factor in the exponent. Our upper bound for product spaces matches the lower bound for hypercubes only up to *polylog* factors of  $d, k, r, 1/\epsilon$  in the exponent. In particular, the upper bound for product spaces goes to  $\infty$  as any one of the parameters  $r, k$ , or  $1/\epsilon$  grow to  $\infty$ , whereas the lower bound for the hypercube can be at most  $\exp(\Theta(d))$  simply because  $|\{0, 1\}^d| = 2^d$  and so any function  $f: \{0, 1\}^d \rightarrow \mathbb{R}$  can be learned *exactly* with  $\exp(O(d))$  samples. It seems intuitive that sample-based testing and learning of  $k$ -monotone functions over  $[n]^d$  should require  $n^{\Omega(d)}$  samples as either of the parameters  $k$  or  $r$  approaches  $\infty$ . A corollary of such a result would be that the sample-complexity of these problems for  $f: \mathbb{R}^d \rightarrow [r]$  grow to  $\infty$  as  $k$  or  $r$  approach  $\infty$ . Moreover, if this is true, then  $k$ -monotonicity of functions  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  is not testable with a finite number of samples. Our results do not address this and it would be interesting to investigate this further.

► **Question 8.** *Is there a lower bound for sample-based  $k$ -monotonicity testing of functions  $f: [n]^d \rightarrow [r]$  which approaches  $n^{\Omega(d)}$  as  $r$  or  $k$  go to  $\infty$ ?*

## 1.4 Related Work

Monotone functions and their generalization to  $k$ -monotone functions have been extensively studied within property testing and learning theory over the last 25 years. We highlight some of the results which are most relevant to our work. Afterwards, we discuss some selected works on sample-based property testing.

### 1.4.1 Sample-Based Monotonicity Testing

Sample-based monotonicity testing of Boolean functions over the hypercube,  $\{0, 1\}^d$ , was considered by [43] (see [43, Theorems 5 and 6]) who gave an upper bound of  $O(\sqrt{2^d/\varepsilon})$  and a lower bound of  $\Omega(\sqrt{2^d/\varepsilon})$  for  $\varepsilon = O(d^{-3/2})$ . Sample-based monotonicity testing over general partial orders was studied by [42] who gave a  $O(\sqrt{N/\varepsilon})$  one-sided error tester for functions  $f: D \rightarrow \mathbb{R}$  where  $D$  is any partial order on  $N$  elements. Sample-based monotonicity testing of functions on the line  $f: [n] \rightarrow [r]$  was studied by [58] who gave a one-sided error upper bound of  $O(\sqrt{r/\varepsilon})$  and a matching lower bound of  $\Omega(\sqrt{r})$  for all sample-based testers.

### 1.4.2 Query-Based Monotonicity Testing

Monotonicity testing has been extensively studied in the standard query model [59, 35, 43, 34, 56, 42, 47, 1, 48, 2, 39, 62, 8, 22, 36, 16, 60, 9, 26, 27, 32, 7, 19, 30, 25, 52, 4, 33, 11, 58, 12, 49, 15, 21, 14, 13, 29]. When discussing these works we treat  $\varepsilon$  as a small constant for brevity. For  $f: \{0, 1\}^d \rightarrow \{0, 1\}$ , the non-adaptive query complexity has been established at  $\tilde{\Theta}(\sqrt{d})$  [53, 33] with an adaptive lower bound of  $\tilde{\Omega}(d^{1/3})$  [33]. This gap for adaptive monotonicity testing of Boolean functions is still an outstanding open question. For  $f: [n]^d \rightarrow \{0, 1\}$  and  $f: \mathbb{R}^d \rightarrow \{0, 1\}$  under product measures, a recent result of [13] established a non-adaptive upper bound of  $d^{1/2+o(1)}$ . For functions  $f: \{0, 1\}^d \rightarrow [r]$ , [15] showed upper and lower bounds of  $\tilde{\Theta}(\min(r\sqrt{d}, d))$  for non-adaptive, one-sided error testers and there is a general (adaptive) lower bound of  $\Omega(\min(d, r^2))$  due to [16]. For real-valued functions  $f: [n]^d \rightarrow \mathbb{R}$ , the query complexity is known to be  $\Theta(d \log n)$ . The upper bound is non-adaptive [26] and the lower bound holds even for adaptive testers [28].

### 1.4.3 $k$ -Monotonicity Testing

The generalization to  $k$ -monotonicity testing has also been studied in the standard query model by [46, 24]. These works show that the query-complexity of non-adaptive one-sided error  $k$ -monotonicity testing is  $\exp(\tilde{\Theta}(\sqrt{d}))$  for all  $k \geq 2$ , demonstrating an interesting separation between (1)-monotonicity and 2-monotonicity.

### 1.4.4 Learning Monotone Functions

Monotone Boolean functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  were studied in the context of learning theory by [23] who showed that they can be (improperly) learned to error  $\varepsilon$  under the uniform distribution with  $\exp(\tilde{O}(\frac{1}{\varepsilon}\sqrt{d}))$  time and samples. Very recent works [54, 55] have given *agnostic proper* learning algorithms with the same complexity.

### 1.4.5 Learning $k$ -Monotone Functions

The result of [23] was generalized by [17] who gave upper and lower bounds of  $\exp(\tilde{\Theta}(\frac{k}{\varepsilon}\sqrt{d}))$  for learning  $k$ -monotone Boolean functions  $f: \{0, 1\}^d \rightarrow \{0, 1\}$ . For Boolean functions over hypergrids  $f: [n]^d \rightarrow \{0, 1\}$ , [24] gave an upper bound of  $\exp(\tilde{O}(\min(\frac{k}{\varepsilon^2}\sqrt{d}, d)))$  where  $\tilde{O}(\cdot)$  hides polylog factors of  $d, k, 1/\varepsilon$ . This result was generalized to functions  $f: \mathbb{R}^d \rightarrow \{0, 1\}$  under product measures by [49].

### 1.4.6 Sample-Based Property Testing

The notion of sample-based property testing was first presented and briefly studied by [44]. Broader studies of sample-based testing and its relationship with query-based testing have since been given by [40, 41, 45]. A characterization of properties which are testable with a constant number of samples was given by [20].

As we mentioned, sample-based algorithms are the standard model in learning theory, and learning requires at least as many samples as testing for every class of functions. Thus, it is natural to ask, when is testing *easier* than learning in terms of sample complexity? This question is referred to as *testing vs learning* and has been studied by [51] and more recently by [18, 37, 38].

There has also been work studying models that interpolate between query-based and sample-based testers. For instance, [3] introduced the notion of *active testing*, where the tester may make queries, but only on points from a polynomial-sized batch of unlabeled samples drawn from the underlying distribution. This was inspired by the notion of *active learning* which considers learning problems under this access model.

Sample-based convexity testing of sets over various domains has also seen some recent attention [31, 5, 6, 10].

### 1.5 Learning Functions with Bounded Image Size: Proof of Theorem 3

In this section we give a short proof showing that the learning algorithm of [17] can be extended in a relatively straightforward manner to functions  $f: \{0, 1\}^d \rightarrow [r]$  by increasing the sample-complexity by a factor of  $r$  in the exponent.

**Proof of Theorem 3.** [17, Theorem 1.4] proved this result for the case of  $r = 2$ . In particular, they show that there is a sample-based learning algorithm which given an arbitrary  $k$ -monotone Boolean function  $f$ , outputs  $h$  such that  $\mathbb{P}_h[d(f, h) > \varepsilon] < \delta$  using  $\ln(1/\delta) \cdot \exp(O(\min\{\frac{rk}{\varepsilon}\sqrt{d}\log d, d\}))$  queries<sup>4</sup> to the example oracle,  $EX(f)$ . We will make use of this result.

For each  $t \in [r]$ , let  $f_t: \{0, 1\}^d \rightarrow \{0, 1\}$  denote the thresholded Boolean function defined as  $f_t(x) := \mathbf{1}(f(x) \geq t)$ . Observe that for all  $x \in \{0, 1\}^d$  we have  $f(x) = \operatorname{argmax}_t \{f_t(x) = 1\}$ . Thus, for each  $t \in [r]$ , run the learning algorithm of [17] with error parameters set to  $\varepsilon' := \varepsilon/r$  and  $\delta = 1/3r$  to obtain a hypothesis  $h_t$ . We have  $\mathbb{P}[d(h_t, f_t) > \varepsilon/r] < 1/3r$ . By a union bound, with probability at least  $2/3$ , every  $t \in [r]$  satisfies  $d(h_t, f_t) \leq \varepsilon/r$ . Moreover, if this holds then by another union bound we have  $\mathbb{P}_x[\exists t \in [r]: h_t(x) \neq f(x)] \leq \varepsilon$ . Thus, the hypothesis  $h(x) := \operatorname{argmax}_t \{h_t(x) = 1\}$  satisfies  $d(h, f) \leq \varepsilon$ . The number of samples used is  $\ln(1/\delta) \cdot \exp(O(\min\{\frac{k}{\varepsilon'}\sqrt{d}\log d, d\})) = \exp(O(\min\{\frac{rk}{\varepsilon}\sqrt{d}\log d, d\}))$  and this completes the proof.  $\blacktriangleleft$

## 2 Preliminaries on $k$ -Monotonicity

We use the notation  $[n] := \{0, 1, \dots, n-1\}$ .

**► Definition 9.** Given a poset  $\mathcal{P} = (\mathcal{X}, \preceq)$  and a function  $f: \mathcal{X} \rightarrow \mathbb{R}$ , an  $m$ -alternating chain is a sequence of points  $x_1 \prec x_2 \prec \dots \prec x_m$  such that for all  $i \in \{1, \dots, m-1\}$ ,

1.  $f(x_{i+1}) - f(x_i) < 0$  when  $i$  is odd, and
2.  $f(x_{i+1}) - f(x_i) > 0$  when  $i$  is even.

<sup>4</sup> Their result (Thm 1.4 of [17]) is stated for constant  $\delta$ , but can be easily extended to arbitrary  $\delta$  with the stated query complexity by replacing Thm 3.1 in their proof with the Low-Degree Algorithm stated for general  $\delta$ .

## 37:10 Sample-Based Testing and Learning of $k$ -Monotone Functions

► **Definition 10** ( $k$ -monotonicity). For a poset  $\mathcal{P} = (\mathcal{X}, \preceq)$ , a function  $f: \mathcal{X} \rightarrow \mathbb{R}$  is called  $k$ -monotone if it does not have any  $(k+1)$ -alternating chains.

Let  $\mathcal{M}_{\mathcal{P},k}$  denote the set of all  $k$ -monotone functions  $f: \mathcal{X} \rightarrow \mathbb{R}$  over the poset  $\mathcal{P} = (\mathcal{X}, \preceq)$ . The Hamming distance between two functions  $f, g: \mathcal{X} \rightarrow \mathbb{R}$  is  $d(f, g) = |\mathcal{X}|^{-1} \cdot |\{x \in \mathcal{X}: f(x) \neq g(x)\}|$ . The distance to  $k$ -monotonicity of  $f$  is denoted by  $\varepsilon(f, \mathcal{M}_{\mathcal{P},k}) := \min_{g \in \mathcal{M}_{\mathcal{P},k}} d(f, g)$ . The following claim is our main tool for lower bounding the distance to  $k$ -monotonicity.

▷ **Claim 11.** Let  $f: \mathcal{X} \rightarrow \mathbb{R}$  and  $k' \geq 3k$  be an integer. Let  $\mathcal{C} \subset \mathcal{X}^{k'}$  be a collection of disjoint  $k'$ -alternating chains for  $f$ . Then

$$\varepsilon(f, \mathcal{M}_{\mathcal{P},k}) \geq \frac{1}{3|\mathcal{X}|} \cdot \left| \bigcup_{C \in \mathcal{C}} C \right|.$$

Proof. Observe that every  $k$ -monotone function  $g \in \mathcal{M}_{\mathcal{P},k}$  has the following property: for every  $C = (x_1, x_2, \dots, x_{k'}) \in \mathcal{C}$ , the sequence

$$(1, g(x_2) - g(x_1), g(x_3) - g(x_2), \dots, g(x_{k'}) - g(x_{k'-1}))$$

changes sign at most  $k-1$  times, whereas the sequence

$$(1, f(x_2) - f(x_1), f(x_3) - f(x_2), \dots, f(x_{k'}) - f(x_{k'-1}))$$

changes sign exactly  $k'-1$  times. We have prepended a 1 so that the first sign change occurs as soon as the function value decreases. Now, changing  $f(x_i)$  can only reduce the number of times the sequence changes sign by at most 2 and so  $|\{i: f(x_i) \neq g(x_i)\}| \geq \frac{k'-k}{2}$ . Summing over all chains in  $\mathcal{C}$  and normalizing yields

$$d(f, g) \geq \frac{k'-k}{2} \cdot \frac{|\mathcal{C}|}{|\mathcal{X}|} \geq \frac{k'}{3} \cdot \frac{|\mathcal{C}|}{|\mathcal{X}|} \geq \frac{1}{3|\mathcal{X}|} \cdot \left| \bigcup_{C \in \mathcal{C}} C \right|$$

where the second inequality follows from  $k \leq k'/3$  and the third inequality is due to the fact that the chains in  $\mathcal{C}$  are all disjoint and each of size  $k'$ . This completes the proof since this inequality holds for all  $g \in \mathcal{M}_{\mathcal{P},k}$ . ◁

We use the notation  $\mathcal{M}_{r,k}$  to denote the set of all  $k$ -monotone functions  $f: \{0, 1\}^d \rightarrow [r]$  over the hypercube whose image has at most  $r$  distinct values.

### 3 Lower Bound for Sample-Based Testers

In this section we prove Theorem 1, our lower bound on the sample-complexity of testing  $k$ -monotonicity of functions  $f: \{0, 1\}^d \rightarrow [r]$ . We refer the reader to Section 1.2.1 for a discussion of our main ideas and a proof sketch for the special case of  $k=1$  and  $r=2$ , i.e. *monotone Boolean* functions. Our proof follows the standard approach of defining a pair of distributions  $\mathcal{D}_{\text{yes}}, \mathcal{D}_{\text{no}}$  over functions  $f: \{0, 1\}^d \rightarrow [r]$  which satisfy the following:

- $\mathcal{D}_{\text{yes}}$  is supported over  $k$ -monotone functions.
- Functions drawn from  $\mathcal{D}_{\text{no}}$  are typically  $\Omega(\varepsilon)$ -far from  $k$ -monotone:  $\mathbb{P}_{f \sim \mathcal{D}_{\text{no}}}[\varepsilon(f, \mathcal{M}_{r,k}) = \Omega(\varepsilon)] = \Omega(1)$ .
- The distributions over labeled examples from  $\mathcal{D}_{\text{yes}}$  and  $\mathcal{D}_{\text{no}}$  are close in TV-distance.

Our construction uses a generalized version of a family functions known as random Talagrand DNFs, which were used by [4] and [33] to prove lower bounds for testing monotonicity of Boolean functions with adaptive and non-adaptive queries.

Let  $r, k$  satisfy  $rk \leq \frac{\varepsilon\sqrt{d}}{24300}$ . For convenience, we will assume that  $\frac{k(r-1)}{\varepsilon}$  and  $\sqrt{d}$  are integers and that  $\frac{k(r-1)}{\varepsilon}$  divides  $\sqrt{d}$ . Let  $L_\ell := \{x \in \{0, 1\}^d : |x| = \ell\}$  denote the  $\ell$ 'th Hamming level of the hypercube. We partition  $\bigcup_{\ell \in [0, \varepsilon\sqrt{d}]} L_{d/2+\ell}$  into  $k(r-1)$  blocks as follows. For each  $i \in [k(r-1)]$ , define

$$B_i = \bigcup_{\ell = i \cdot \frac{\varepsilon\sqrt{d}}{k(r-1)}}^{(i+1) \cdot \frac{\varepsilon\sqrt{d}}{k(r-1)} - 1} L_{\frac{d}{2} + \ell}.$$

The idea of our proof is to define a random DNF within each  $B_i$ . The *width* of each DNF will be set to  $w := \frac{(r-1)k\sqrt{d}}{2\varepsilon}$  and for each  $i$ , the number of terms in the DNF within  $B_i$  will be set to  $N_i := 2^w \cdot e^{-i} = 2^{\frac{(r-1)k\sqrt{d}}{2\varepsilon}(1-o(1))}$ . The DNF defined over  $B_i$  will assign function values from  $\{i \bmod (r-1), i \bmod (r-1) + 1\}$ . The terms in each DNF will be chosen randomly from the following distribution. We think of terms as points  $t \in \{0, 1\}^d$  in the hypercube where another point  $x$  *satisfies*  $t$  if  $t \preceq x$ , i.e.  $t_i = 1$  implies  $x_i = 1$ .

► **Definition 12** (Term distribution). *A term  $t \in \{0, 1\}^d$  is sampled from the distribution  $\mathcal{D}_{\text{term}}$  as follows. Form a (multi)-set  $S \subseteq [d]$  by choosing  $w$  independent uniform samples from  $[d]$ . For each  $a \in [d]$ , let  $t_a := \mathbf{1}(a \in S)$ .*

### 3.1 The Distributions $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$

We now define the yes and no distributions over functions  $f: \{0, 1\}^d \rightarrow [r]$ . For each  $i \in [k(r-1)]$ , choose terms  $t^{i,1}, \dots, t^{i,N_i}$  i.i.d. from  $\mathcal{D}_{\text{term}}$  and let  $\mathbf{t} = \{t^{i,j} : i \in [k(r-1)], j \in [N_i]\}$  denote the random set of all terms. Now, for each  $i \in [k(r-1)]$  and  $j \in [N_i]$ , define the set

$$U_{i,j} = \left\{ x \in B_i : x \succeq t^{i,j} \text{ and } x \not\succeq t^{i,j'} \text{ for all } j' \neq j \right\} \quad (1)$$

of all points in the  $i$ 'th block that satisfy the  $j$ 'th term *uniquely*. Let  $U_i = \bigcup_{j \in [N_i]} U_{i,j}$  denote the set of points in  $B_i$  that satisfy a unique term. The following claim is key to our result and motivates our choice of  $w$  and  $N_i$ . We defer its proof to Section 3.2.

► **Claim 13.** For any  $i \in [k(r-1)]$ ,  $j \in [N_i]$ , and  $x \in B_i$ , we have

$$\frac{1}{45N_i} \leq \mathbb{P}_{\mathbf{t}}[x \in U_{i,j}] \leq \frac{3}{N_i}.$$

As a corollary, we have  $\mathbb{P}_{\mathbf{t}}[x \in U_i] \geq 1/45$ .

Functions drawn from  $\mathcal{D}_{\text{yes}}$  are generated as follows. For each  $i \in [k(r-1)]$  choose a uniform random assignment

$$\phi_i: [N_i] \rightarrow \{i \bmod (r-1), i \bmod (r-1) + 1\} \text{ and let } \phi = (\phi_i : i \in [k(r-1)]).$$

For every  $x \in B_i$  define

$$f_{\mathbf{t},\phi}(x) = \begin{cases} i \bmod (r-1), & \text{if } \forall j \in [N_i], x \not\succeq t^{i,j} \\ i \bmod (r-1) + 1, & \text{if } \exists j \neq j' \in [N_i], x \succeq t^{i,j}, t^{i,j'} \\ \phi_i(j), & \text{if } x \in U_{i,j}. \end{cases}$$

## 37:12 Sample-Based Testing and Learning of $k$ -Monotone Functions

Functions drawn  $\mathcal{D}_{\text{no}}$  are generated as follows. For each  $i \in [k(r-1)]$  choose a uniform random function

$$\mathbf{r}_i: U_i \rightarrow \{i \bmod (r-1), i \bmod (r-1) + 1\} \text{ and let } \mathbf{r} = (\mathbf{r}_i: i \in [k(r-1)]).$$

For each  $x \in B_i$  define

$$f_{\mathbf{t}, \mathbf{r}}(x) = \begin{cases} i \bmod (r-1), & \text{if } \forall j \in [N_i], x \not\succeq t^{i,j} \\ i \bmod (r-1) + 1, & \text{if } \exists j \neq j' \in [N_i], x \succeq t^{i,j}, t^{i,j'} \\ \mathbf{r}_i(x), & \text{if } x \in U_i. \end{cases}$$

For  $x$  not belonging to any  $B_i$ : if  $|x| < \frac{d}{2}$ , then both the yes and no distributions assign value 0 and if  $|x| \geq \frac{d}{2} + \varepsilon\sqrt{d}$ , then both the yes and no distributions assign value  $r-1$ .

In summary, a function  $f_{\mathbf{t}, \phi} \sim \mathcal{D}_{\text{yes}}$  assigns the same random value  $\phi_i(j) \in \{i \bmod (r-1), i \bmod (r-1) + 1\}$  to all points in  $U_{i,j}$ , which results in a  $k$ -monotone function, whereas a function  $f_{\mathbf{t}, \mathbf{r}} \sim \mathcal{D}_{\text{no}}$  assigns an i.i.d. uniform random  $\{i \bmod (r-1), i \bmod (r-1) + 1\}$ -value to each point in  $U_i$ , resulting in a function that is far from being  $k$ -monotone. By construction, to detect any difference between these cases a tester will need to sample at least two points from the same  $U_{i,j}$ . Theorem 1 follows immediately from the following three lemmas.

► **Lemma 14.** *Every function in the support of  $\mathcal{D}_{\text{yes}}$  is  $k$ -monotone.*

**Proof.** Consider any  $f_{\mathbf{t}, \phi}(x) \in \text{supp}(\mathcal{D}_{\text{yes}})$ . For each  $a \in [k]$ , consider the union of  $r-1$  blocks formed by

$$Y_a := B_{a(r-1)} \cup B_{a(r-1)+1} \cup \dots \cup B_{(a+1)(r-1)-1}.$$

Recall that if  $|x| < d/2$ , then  $f_{\mathbf{t}, \phi}(x) = 0$  and if  $|x| \geq d/2 + \varepsilon\sqrt{d}$ , then  $f_{\mathbf{t}, \phi}(x) = r-1$ . If  $d/2 \leq |x| < d/2 + \varepsilon\sqrt{d}$ , then  $x \in \bigcup_{a \in [k]} Y_a$ . Therefore, it suffices to show that for any pair of comparable points  $x \prec y \in Y_a$ , we have  $f_{\mathbf{t}, \phi}(x) \leq f_{\mathbf{t}, \phi}(y)$ . Firstly, observe that by construction all points  $z \in B_{a(r-1)+b}$  have function value  $f_{\mathbf{t}, \phi}(z) \in \{b, b+1\}$ . Since  $x \prec y$ , if  $x$  and  $y$  are in different blocks, then  $x \in B_{a(r-1)+b}$  and  $y \in B_{a(r-1)+b'}$  where  $b < b'$  and so the inequality is satisfied. Therefore, we may assume  $x, y \in B_{a(r-1)+b}$  are in the same block. Since  $x \prec y$ , if  $t \prec x$  for some term  $t \in \text{supp}(\mathcal{D}_{\text{term}})$ , then  $t \prec y$  as well. I.e. the set of terms in  $B_{a(r-1)+b}$  satisfied by  $y$  is a superset of the set of terms in  $B_{a(r-1)+b}$  satisfied by  $x$ . By construction, this implies  $f_{\mathbf{t}, \phi}(x) \leq f_{\mathbf{t}, \phi}(y)$ . ◀

► **Lemma 15.** *For  $f_{\mathbf{t}, \mathbf{r}} \sim \mathcal{D}_{\text{no}}$ , we have  $\mathbb{P}_{\mathbf{t}, \mathbf{r}}[\varepsilon(f_{\mathbf{t}, \mathbf{r}}, \mathcal{M}_{r,k}) = \Omega(\varepsilon)] = \Omega(1)$ .*

We prove Lemma 15 in Section 3.4.

► **Lemma 16.** *Given a collection of points  $\mathbf{x} = (x_1, \dots, x_s) \in (\{0, 1\}^d)^s$  and a function  $f: \{0, 1\}^d \rightarrow [r]$ , let  $(\mathbf{x}, f(\mathbf{x})) = ((x_1, f(x_1)), \dots, (x_s, f(x_s)))$  denote the corresponding collection of labelled examples. Let  $\mathcal{E}_{\text{yes}}$  and  $\mathcal{E}_{\text{no}}$  denote the distributions over  $(\mathbf{x}, f(\mathbf{x}))$  when  $\mathbf{x}$  consists of  $s$  i.i.d. uniform samples and  $f \sim \mathcal{D}_{\text{yes}}$  and  $f \sim \mathcal{D}_{\text{no}}$ , respectively. If  $s \leq 2^{\frac{(r-1)k\sqrt{d}}{5\varepsilon}}$ , then the total variation distance between  $\mathcal{E}_{\text{yes}}$  and  $\mathcal{E}_{\text{no}}$  is  $o(1)$ .*

We prove Lemma 16 in Section 3.3.

### 3.2 Proof of Claim 13

Proof. Recall  $w = \frac{(r-1)k\sqrt{d}}{2\varepsilon}$ ,  $N_i = 2^w \cdot e^{-i}$ , the definition of  $\mathcal{D}_{\text{term}}$  from Definition 12, and the definition of  $U_{i,j}$  from Equation (1). Since  $x \in B_i$  we have  $|x| = \frac{d}{2} + \ell$  where  $\frac{i\varepsilon\sqrt{d}}{k(r-1)} \leq \ell < \frac{(i+1)\varepsilon\sqrt{d}}{k(r-1)}$ . Note that  $\mathbb{P}_{t \sim \mathcal{D}_{\text{term}}}[t \preceq x] = (|x|/d)^w$  since  $t \preceq x$  iff the non-zero coordinates of  $t$  are a subset of the non-zero coordinates of  $x$ . Therefore, we have

$$\mathbb{P}_{\mathbf{t}}[x \in U_{i,j}] = \mathbb{P}_{t^{i,j}}[t^{i,j} \preceq x] \cdot \prod_{j' \in [N_i] \setminus \{j\}} \mathbb{P}_{t^{i,j'}}[t^{i,j'} \not\preceq x] = (|x|/d)^w (1 - (|x|/d)^w)^{N_i - 1}.$$

Note that the first term is upper bounded as

$$(|x|/d)^w \leq \left( \frac{\frac{d}{2} + \frac{(i+1)\varepsilon\sqrt{d}}{k(r-1)}}{d} \right)^w = \frac{1}{2^w} \left( 1 + \frac{2\varepsilon}{k(r-1)\sqrt{d}} \cdot (i+1) \right)^w \leq \frac{e^{i+1+o(1)}}{2^w} \leq \frac{e^{1+o(1)}}{N_i}$$

and this immediately implies the upper bound on  $\mathbb{P}_{\mathbf{t}}[x \in U_{i,j}]$ . We can also lower bound this quantity by

$$(|x|/d)^w \geq \left( \frac{\frac{d}{2} + \frac{i\varepsilon\sqrt{d}}{k(r-1)}}{d} \right)^w = \frac{1}{2^w} \left( 1 + \frac{2\varepsilon}{k(r-1)\sqrt{d}} \cdot i \right)^w \geq \frac{e^{i-o(1)}}{2^w} \geq \frac{1}{e^{o(1)}N_i}.$$

Now, combining our upper and lower bounds on  $(|x|/d)^w$  yields

$$\mathbb{P}_{\mathbf{t}}[x \in U_{i,j}] \geq \frac{1}{e^{o(1)}N_i} \left( 1 - \frac{e^{1+o(1)}}{N_i} \right)^{N_i} \geq \frac{1}{e^{o(1)}N_i} e^{-(1+o(1)) \cdot e^{1+o(1)}} \geq \frac{1}{e^{e+1}N_i} \geq \frac{1}{45N_i}.$$

◁

### 3.3 $\mathcal{D}_{\text{yes}}$ and $\mathcal{D}_{\text{no}}$ are Hard to Distinguish: Proof of Lemma 16

**Proof.** Recall the definition of the set  $U_{i,j}$  in Equation (1). For  $a \neq b \in [s]$ , let  $E_{ab}$  denote the event that  $x_a$  and  $x_b$  belong to the same  $U_{i,j}$  for some  $i \in [k(r-1)]$  and  $j \in [N_i]$ . Observe that conditioned on  $\bigvee_{a,b} E_{ab}$ , the distributions  $\mathcal{E}_{\text{yes}}$  and  $\mathcal{E}_{\text{no}}$  are identical. Let  $x, y \in \{0, 1\}^d$  denote two i.i.d. uniform samples. We have

$$\begin{aligned} \mathbb{P}[E_{ab}] &= \mathbb{P}_{x,y,\mathbf{t}} \left[ \bigvee_{i,j} (x \in U_{i,j} \wedge y \in U_{i,j}) \right] \\ &= \sum_{i,j} \mathbb{P}_{x,y,\mathbf{t}} [x \in U_{i,j} \wedge y \in U_{i,j}] = \sum_{i,j} \mathbb{P}_{x,\mathbf{t}} [x \in U_{i,j}]^2 \end{aligned} \quad (2)$$

where the first step holds since the  $U_{i,j}$ 's are disjoint and the second step holds by independence of  $x$  and  $y$ . Now, for a fixed  $i \in [k(r-1)]$  and  $j \in [N_i]$  we have the following: by Claim 13, for  $x \in B_i$  we have  $\mathbb{P}_{\mathbf{t}}[x \in U_{i,j}] \leq \frac{3}{N_i}$  and for  $x \notin B_i$  we have  $\mathbb{P}_{\mathbf{t}}[x \in U_{i,j}] = 0$ . Therefore  $\mathbb{P}_{x,\mathbf{t}}[x \in U_{i,j}] \leq \frac{3}{N_i}$ . Therefore, the RHS of Equation (2) is bounded as

$$\sum_{i,j} \mathbb{P}_{x,\mathbf{t}} [x \in U_{i,j}]^2 \leq \sum_i N_i \cdot \mathbb{P}_{x,\mathbf{t}} [x \in U_{i,j}]^2 \leq \sum_i \frac{9}{N_i} \leq rk \cdot \frac{9}{N_{k(r-1)-1}}$$

since the  $N_i$ 's are decreasing with respect to  $i$ . Therefore,

$$d_{TV}(\mathcal{E}_{\text{yes}}, \mathcal{E}_{\text{no}}) \leq \mathbb{P}_{\mathbf{t}} \left[ \bigvee_{a,b \in [s]} E_{ab} \right] \leq s^2 \cdot rk \cdot \frac{9}{N_{k(r-1)-1}} = o(1)$$

since  $N_{k(r-1)-1} = 2^{\frac{(r-1)k\sqrt{d}}{2\varepsilon}} (1-o(1)) = \omega(s^2 \cdot rk)$ .

◀

### 3.4 Functions Drawn from $\mathcal{D}_{\text{no}}$ are Far from $k$ -Monotone: Proof of Lemma 15

**Proof.** We will use Claim 11, restated below for the special case of  $r$ -valued functions over the hypercube. Recall that  $\mathcal{M}_{r,k}$  is the set of  $k$ -monotone functions  $f: \{0,1\}^d \rightarrow [r]$ .

▷ **Claim 17.** Let  $f: \{0,1\}^d \rightarrow [r]$  and  $k' \geq 3k$  be an integer. Let  $\mathcal{C} \subset (\{0,1\}^d)^{k'}$  be a collection of disjoint  $k'$ -alternating chains for  $f$ . Then

$$\varepsilon(f, \mathcal{M}_{r,k}) \geq \frac{1}{3 \cdot 2^d} \cdot \left| \bigcup_{C \in \mathcal{C}} C \right|.$$

From the above claim, we can lower bound the distance to  $k$ -monotonicity of  $f$  by showing that it contains a collection of disjoint  $k'$ -alternating chains where  $k' \geq 3k$  whose union makes up an  $\Omega(\varepsilon)$ -fraction of the hypercube.

Recall  $U_i = U_{i,1} \cup \dots \cup U_{i,N_i} \subseteq B_i$  and note that  $f_{\mathbf{t},\mathbf{r}} \sim \mathcal{D}_{\text{no}}$  takes values only from  $\{i \bmod (r-1), i \bmod (r-1) + 1\}$  in  $B_i$ . In particular, for  $a \in \{0, 1, \dots, r-2\}$ , let

$$X_a = B_a \cup B_{(r-1)+a} \cup B_{2(r-1)+a} \cup \dots \cup B_{(k-1)(r-1)+a} = \bigcup_{i \in [k]} B_{i(r-1)+a} \quad (3)$$

and note that all points  $x \in X_a$  are assigned value  $f_{\mathbf{t},\mathbf{r}}(x) \in \{a, a+1\}$ . Moreover, this value is chosen uniformly at random when  $x \in \bigcup_{i \in [k]} U_{i(r-1)+a}$ , which occurs with probability  $\geq 1/45$  by Claim 13. Let  $k'' := \frac{\varepsilon\sqrt{d}}{r-1}$  and recall that we are assuming  $rk \leq \frac{\varepsilon\sqrt{d}}{24300}$  and so  $k'' \geq 24300k$ . We first show there exists a large collection  $\mathcal{C}_a$  of length- $k''$  disjoint chains in  $X_a$  for all  $a \in \{0, 1, \dots, r-2\}$ .

▷ **Claim 18.** For every  $a \in \{0, 1, \dots, r-2\}$ , there exists a collection of vertex disjoint chains  $\mathcal{C}_a \subset (X_a)^{k''}$  in  $X_a$  of length  $k''$  of size  $|\mathcal{C}_a| \geq \Omega(\frac{2^d}{\sqrt{d}})$ .

**Proof.** We start by showing that there is a large matching in the transitive closure of the hypercube from  $L_{\frac{d}{2}}$  to  $L_{\frac{d}{2}+\varepsilon\sqrt{d}-1}$ . Consider the bipartite graph  $(U, V, E)$  where  $U := L_{\frac{d}{2}}$ ,  $V := L_{\frac{d}{2}+\varepsilon\sqrt{d}-1}$ , and  $E := \{(x, y) \in U \times V: x \prec y\}$ . Observe that vertices in  $U$  have degree exactly  $\Delta := \binom{\frac{d}{2}}{\varepsilon\sqrt{d}-1}$  while vertices in  $V$  have degree exactly  $\binom{\frac{d}{2}+\varepsilon\sqrt{d}-1}{\varepsilon\sqrt{d}-1} \geq \Delta$ . Note also that  $|V| = \binom{\frac{d}{2}+\varepsilon\sqrt{d}-1}{\varepsilon\sqrt{d}-1} \geq \Omega(\frac{2^d}{\sqrt{d}})$  by Stirling's approximation. We now use the following claim from [10].

▷ **Claim 19 (Claim 5.10 of [10]).** Let  $(U, V, E)$  be a bipartite graph and  $\Delta > 0$  be such that (a) each vertex  $x \in U$  has degree exactly  $\Delta$  and (b) each vertex  $y \in V$  has degree at least  $\Delta$ . Then there exists a matching  $M \subseteq E$  in  $(U, V, E)$  of size  $|M| \geq \frac{1}{2}|V|$ .

By the above claim and the previous observations, there exist subsets  $S \subseteq L_{\frac{d}{2}}$  and  $T \subseteq L_{\frac{d}{2}+\varepsilon\sqrt{d}-1}$  of size  $|S| = |T| = \Omega(\frac{2^d}{\sqrt{d}})$  and a bijection  $\phi: S \rightarrow T$  satisfying  $x \prec \phi(x)$  for all  $x \in S$ . We now use the following routing theorem due to Lehman and Ron to obtain a collection of disjoint chains from  $S$  to  $T$ .

► **Theorem 20 (Lehman-Ron, [56]).** Let  $a < b$  and  $S \subseteq L_a$ ,  $T \subseteq L_b$  where  $m := |S| = |T|$ . Moreover, suppose there is a bijection  $\phi: S \rightarrow T$  satisfying  $x \prec \phi(x)$  for all  $x \in S$ . Then there exist  $m$  vertex disjoint paths from  $S$  to  $T$  in the hypercube.



Now, invoking the above theorem on our bijection  $\phi: S \rightarrow T$  yields a collection  $P$  of  $|P| \geq \Omega(\frac{2^d}{\sqrt{d}})$  vertex disjoint paths from  $L_{\frac{d}{2}}$  to  $L_{\frac{d}{2} + \varepsilon\sqrt{d}-1}$ . For each  $a \in \{0, 1, \dots, r-2\}$ , let  $\mathcal{C}_a$  denote the collection of chains formed by taking a path in  $P$  and including only the vertices from  $X_a$  (recall Equation (3)). Note that the resulting chains in  $\mathcal{C}_a$  are of length  $k'' = \frac{\varepsilon\sqrt{d}}{r-1}$ . This completes the proof of Claim 18.  $\blacktriangleleft$

From Claim 18, we have  $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{r-2}$  where each  $\mathcal{C}_a \subset (X_a)^{k''}$  is a collection of vertex disjoint chains of length  $k'' \geq 24300k$  of size  $|\mathcal{C}_a| \geq \Omega(\frac{2^d}{\sqrt{d}})$ . Fix a chain  $C = (x_1, x_2, \dots, x_{k''}) \in \mathcal{C}_a$ . Let  $A(C)$  be the random variable which denotes the max-length alternating sub-chain (recall Definition 9) of  $C$  over a random  $f_{\mathbf{t}, \mathbf{r}} \sim \mathcal{D}_{\text{no}}$ . Fix  $x_j$  in the chain and suppose  $x_j \in B_i \subseteq X_a$ . By Claim 13,  $\mathbb{P}_{\mathbf{t}}[x_j \in U_i] \geq 1/45$ . Moreover, conditioned on  $x_j \in U_i$ ,  $f_{\mathbf{t}, \mathbf{r}}(x_j)$  is chosen from  $\{a, a+1\}$  uniformly at random. Thus, any step of the sequence

$$(1, f_{\mathbf{t}, \mathbf{r}}(x_2) - f_{\mathbf{t}, \mathbf{r}}(x_1), f_{\mathbf{t}, \mathbf{r}}(x_3) - f_{\mathbf{t}, \mathbf{r}}(x_2), \dots, f_{\mathbf{t}, \mathbf{r}}(x_{k''}) - f_{\mathbf{t}, \mathbf{r}}(x_{k''-1}))$$

is non-zero *and* differs in sign from the previous non-zero step with probability at least  $1/90$  and so  $\mathbb{E}[A(C)] \geq k''/90$ . I.e.,  $0 \leq \mathbb{E}[k'' - A(C)] < k''(1 - \frac{1}{90})$ . Thus, using Markov's inequality we have

$$\mathbb{P}\left[A(C) < \frac{k''}{8100}\right] = \mathbb{P}\left[k'' - A(C) > k''\left(1 - \frac{1}{90}\right)\left(1 + \frac{1}{90}\right)\right] \leq \frac{1}{(1 + \frac{1}{90})} = 1 - \frac{1}{91}. \quad (4)$$

Now, let  $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \dots \cup \mathcal{C}_{r-2}$  and let  $Z := |\{C \in \mathcal{C} : A(C) \geq \frac{k''}{8100}\}|$ . By Equation (4) we have  $\mathbb{E}[Z] \geq |\mathcal{C}|/91$  and  $0 \leq \mathbb{E}[|\mathcal{C}| - Z] \leq |\mathcal{C}|(1 - \frac{1}{91})$ . Again using Markov's inequality, we have

$$\mathbb{P}\left[Z < \frac{|\mathcal{C}|}{8281}\right] = \mathbb{P}\left[|\mathcal{C}| - Z > |\mathcal{C}|\left(1 - \frac{1}{91}\right)\left(1 + \frac{1}{91}\right)\right] \leq \frac{1}{(1 + \frac{1}{91})} = 1 - \frac{1}{92}. \quad (5)$$

Now, for  $C \in \mathcal{C}$  such that  $A(C) \geq k''/8100$ , let  $C'$  be any  $(k''/8100)$ -alternating sub-chain of  $C$ . Let  $\mathcal{C}' = \{C' : C \in \mathcal{C} \text{ such that } A(C) \geq k''/8100\}$  which is a collection of disjoint  $(k''/8100)$ -alternating chains for  $f_{\mathbf{t}, \mathbf{r}}$ . Now, recall that  $k'' \geq 24300k$  and so  $k''/8100 \geq 3k$ . Thus, if  $Z \geq |\mathcal{C}|/8281$ , then  $|\mathcal{C}'| \geq |\mathcal{C}|/8281$  and so by Claim 17 we have

$$\varepsilon(f_{\mathbf{t}, \mathbf{r}}, \mathcal{M}_{r, k}) \geq \frac{1}{3 \cdot 2^d} \left| \bigcup_{C' \in \mathcal{C}'} C' \right| \geq \frac{1}{3 \cdot 2^d} \cdot |\mathcal{C}'| \cdot \frac{k''}{8100} \geq \frac{k'' \cdot |\mathcal{C}|}{201, 228, 300 \cdot 2^d} \quad (6)$$

By Claim 18 we have  $|\mathcal{C}| \geq (r-1) \cdot \Omega(\frac{2^d}{\sqrt{d}})$  and recall that  $k'' = \frac{\varepsilon\sqrt{d}}{r-1}$ . Thus, the RHS of Equation (6) is  $\Omega(\varepsilon)$ . In conclusion,

$$\mathbb{P}_{\mathbf{t}, \mathbf{r}}[\varepsilon(f_{\mathbf{t}, \mathbf{r}}, \mathcal{M}_{r, k}) \geq \Omega(\varepsilon)] \geq \mathbb{P}\left[Z \geq \frac{|\mathcal{C}|}{8281}\right] \geq \frac{1}{92}$$

by Equation (5) and this completes the proof of Lemma 15.  $\blacktriangleleft$

## 4 Learning Upper Bound over Product Spaces

In this section we prove Theorem 6, our upper bound for learning measurable  $k$ -monotone functions in  $\mathbb{R}^d$ . We restate the theorem below without any hidden logarithmic factors and for the case of  $r = 2$ . The theorem for general  $r \geq 2$  can then be obtained by replacing  $\varepsilon$  with  $\varepsilon/r$  and  $\delta$  by  $1/3r$  following the same approach we used to prove Theorem 3 in Section 1.5.

## 37:16 Sample-Based Testing and Learning of $k$ -Monotone Functions

► **Theorem 21.** *Given an arbitrary product measure  $\mu$ , there is a learning algorithm under  $\mu$  which learns any measurable  $k$ -monotone function  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  to error  $\varepsilon$  with probability  $1 - \delta$  with time and sample complexity*

$$\ln\left(\frac{1}{\delta}\right) \cdot \min\left\{\left(d \log(dk/\varepsilon)\right)^{O\left(\frac{k}{\varepsilon} \sqrt{d \log(dk/\varepsilon)}\right)}, \left(\frac{dk}{\varepsilon}\right)^{O(d)}\right\} \quad (7)$$

Our proof uses downsampling to reduce our learning problem over  $\mathbb{R}^d$  to learning over a hypergrid,  $[N]^d$ , under the uniform distribution with mild label noise. In Section 4.1 we synthesize the results from [49] which we borrow for our proof. In Section 4.2 we give two learning results for hypergrids whose time complexities correspond to the two arguments inside the min expression in Equation (7). In Section 4.3 we describe the learning algorithm and prove its correctness.

Throughout this section, let  $\mu = \prod_{i=1}^d \mu_i$  be any product measure over  $\mathbb{R}^d$  and let  $N$  be a power of two satisfying  $8kd/\varepsilon \leq N \leq 16kd/\varepsilon$ .

### 4.1 Reduction to Hypergrids via Downsampling

The idea of downsampling is to construct a grid-partition of  $\mathbb{R}^d$  into  $N^d$  blocks such that (a) the measure of each block under  $\mu$  is roughly  $N^{-d}$ , and (b) the function  $f$  we're trying to learn is constant on most of the blocks. Roughly speaking, this allows us to learn  $f$  under  $\mu$  by learning a proxy for  $f$  over  $[N]^d$  under the uniform distribution. The value of  $N$  needed to achieve this depends on what [49] call the “block boundary size” of the function. Formally, the downsampling procedure constructs query access to maps  $\text{block}: \mathbb{R}^d \rightarrow [N]^d$  and  $\text{blockpoint}: [N]^d \rightarrow \mathbb{R}^d$  which have various good properties which we will spell out in the rest of this section. One should think of  $\text{block}$  as mapping each point  $x \in \mathbb{R}^d$  to the block of the grid-partition that  $x$  belongs to and  $\text{blockpoint}$  as mapping each block to some specific point contained in the block. See [49, Def 2.1] for a formal definition. Given these maps and a function  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  we define the function  $f^{\text{block}}: [N]^d \rightarrow \{\pm 1\}$  as  $f^{\text{block}}(z) = f(\text{blockpoint}(z))$ . We let  $\text{block}(\mu)$  denote the distribution over  $[N]^d$  induced by sampling  $x \sim \mu$  and then taking  $\text{block}(x)$ .

► **Proposition 22** (Downsampling, [49]). *Let  $f: \mathbb{R}^d \rightarrow \{0, 1\}$  be a  $k$ -monotone function and  $N, Q \in \mathbb{Z}^+$ . Using*

$$m := O\left(\frac{NQ^2 d^2}{\min(\delta, \varepsilon)^2} \ln\left(\frac{Nd}{\delta}\right)\right)$$

*samples from  $\mu = \mu_1 \times \dots \times \mu_d$ , there is a downsampling procedure that constructs query access to maps  $\text{block}: \mathbb{R}^d \rightarrow [N]^d$  and  $\text{blockpoint}: [N]^d \rightarrow \mathbb{R}^d$  such that with probability at least  $1 - \delta$  over the random samples, the following two conditions are satisfied:*

1.  $\|\text{block}(\mu) - \text{unif}([N]^d)\|_{TV} \leq \frac{\delta}{Q}$ .
2.  $\mathbb{P}_{x \sim \mu} [f(x) \neq f^{\text{block}}(\text{block}(x))] \leq \varepsilon$ .

*The total running time and number of samples is  $O(m)$ .*

**Proof.** [49, Prop. 2.5] shows that there is a randomized procedure using  $m$  samples from  $\mu$  and  $O(m)$  time which constructs the maps  $\text{block}$  and  $\text{blockpoint}$  such that with probability 1, we get

$$\mathbb{P}_{x \sim \mu} [f(x) \neq f^{\text{block}}(\text{block}(x))] \leq N^{-d} \cdot \text{bbs}(f, N) + \|\text{block}(\mu) - \text{unif}([N]^d)\|_{TV} \quad (8)$$

where  $\text{bbs}(f, N)$  is the  $N$ -block boundary size of  $f$  [49, Def. 2.4], which is at most  $kdN^{d-1}$  when  $f$  is  $k$ -monotone [49, Lemma 7.1]. Thus, the first of the two quantities in the RHS is at most  $kd/N$  which is at most  $\varepsilon/8$  using our definition of  $N$ . Then, [49, Lemma 2.7] states that

$$\mathbb{P} \left[ \left\| \text{block}(\mu) - \text{unif}([N]^d) \right\|_{\text{TV}} > \beta \right] \leq 4Nd \cdot \exp \left( -\frac{\beta^2 m}{18Nd^2} \right) \quad (9)$$

and so invoking this lemma with  $\beta := \min(\delta/4Q, \varepsilon/8)$  and  $m := \frac{18Nd^2}{\beta^2} \ln \left( \frac{16Nd}{\delta} \right)$  completes the proof.  $\blacktriangleleft$

## 4.2 Learning over Hypergrids

For a function  $f: \mathcal{X} \rightarrow \{\pm 1\}$  and a measure  $\mu$  over  $\mathcal{X}$ , recall that the *example oracle* for  $f$  under  $\mu$ , denoted by  $EX(f, \mu)$ , when queried, generates an example,  $(x, f(x))$ , where  $x$  is sampled from  $\mu$ . Given a *noise parameter*  $\eta$ , the *noisy example oracle*  $EX^\eta(f, \mu)$ , when queried, samples  $x$  from  $\mu$ , returns the true example  $(x, f(x))$  with probability  $1 - \eta$ , and returns the corrupted example  $(x, -f(x))$  with probability  $\eta$ . This is referred to as *random classification noise* (RCN).

We prove the following two upper bounds for learning over hypergrids under RCN. The bound in Lemma 23 is relatively straightforward to prove using coupon collector arguments plus some additional work to handle the label noise. We give a proof in the appendix (see Section B in the full version of the paper).

► **Lemma 23** (Coupon Collecting Learner). *Let  $\varepsilon, \delta \in (0, 1)$ ,  $\eta \in (0, 1/2)$ , and  $N \in \mathbb{Z}^+$ . There is an algorithm which, given any  $k$ -monotone function  $f: [N]^d \rightarrow \{\pm 1\}$ , uses at most*

$$\tilde{O} \left( \frac{1}{(1-2\eta)^2} \left( \log \frac{1}{\varepsilon} + \log \frac{1}{\delta} \right) \right) \cdot N^{O(d)}$$

*examples from  $EX^\eta(f, \text{unif}([N]^d))$  and returns  $h: [N]^d \rightarrow \{\pm 1\}$ , satisfying  $\mathbb{P}_h[d(f, h) \leq \varepsilon] \geq 1 - \delta$ .*

► **Lemma 24** (Hypercube Mapping Learner). *Let  $\varepsilon, \delta \in (0, 1)$ ,  $\eta \in (0, 1/2)$ , and  $N \in \mathbb{Z}^+$  be a power of two. There is an algorithm which, given any  $k$ -monotone function  $f: [N]^d \rightarrow \{\pm 1\}$ , uses at most*

$$O \left( \frac{1}{\varepsilon^2(1-2\eta)^2} + \log \frac{1}{\delta} \right) (d \log N)^{O\left(\frac{k}{\varepsilon} \sqrt{d \log N}\right)}$$

*examples from  $EX^\eta(f, \text{unif}([N]^d))$  and returns  $h: [N]^d \rightarrow \{\pm 1\}$ , satisfying  $\mathbb{P}_h[d(f, h) \leq \varepsilon] \geq 1 - \delta$ .*

**Proof.** Let  $b: [N] \rightarrow \{\pm 1\}^{\log N}$  denote the bijection which maps each element of  $[N]$  to its bit representation. Let  $\mathbf{b}: [N]^d \rightarrow \{\pm 1\}^{d \log N}$  be defined as  $\mathbf{b}(x) = (b(x_1), \dots, b(x_d))$ . Given  $f: [N]^d \rightarrow \{\pm 1\}$  define the function  $f^{\text{cube}}: \{\pm 1\}^{d \log N} \rightarrow \{\pm 1\}$  as  $f^{\text{cube}}(z) = f(\mathbf{b}^{-1}(z))$ .

► **Observation 25.** *If  $f$  is  $k$ -monotone over  $[N]^d$ , then  $f^{\text{cube}}$  is  $k$ -monotone over  $\{\pm 1\}^{d \log N}$ .*

**Proof.** Observe that if  $\mathbf{b}(x) \prec \mathbf{b}(y)$  in  $\{\pm 1\}^{d \log N}$ , then  $x \prec y$  in  $[N]^d$ . Thus, if  $\mathbf{b}(x_1) \prec \dots \prec \mathbf{b}(x_m)$  is an  $m$ -alternating chain for  $f^{\text{cube}}$ , then  $x_1 \prec \dots \prec x_m$  is an  $m$ -alternating chain for  $f$ . Therefore, if  $f^{\text{cube}}$  is not  $k$ -monotone, then neither is  $f$ .  $\blacktriangleleft$

## 37:18 Sample-Based Testing and Learning of $k$ -Monotone Functions

Now, given Observation 25 and the bijection  $\mathbf{b}: [N]^d \rightarrow \{\pm 1\}^{d \log N}$ , it suffices to provide a learning algorithm for  $f^{\text{cube}}$ . This is achieved using the Low-Degree Algorithm introduced by [57] which was shown by [50] to be robust to classification noise. Formally, we use the following theorem, which we prove in the appendix for the sake of completeness (see Section A in the full version of the paper).

► **Theorem 26** (Low-Degree Algorithm with Classification Noise). *Let  $\varepsilon, \delta \in (0, 1)$  and  $\eta \in (0, 1/2)$ . Suppose  $\mathcal{C}$  is a concept class of Boolean functions over  $\{\pm 1\}^d$  such that for some fixed positive integer  $\tau$ , all  $f \in \mathcal{C}$  satisfy  $\sum_{S \subseteq [d]: |S| > \tau} \widehat{f}(S)^2 \leq \varepsilon/2$ . Then there is an algorithm  $\mathcal{A}$  which, on any input  $f \in \mathcal{C}$ , uses at most*

$$O\left(\left(\frac{1}{\varepsilon^2(1-2\eta)^2} + \log \frac{1}{\delta}\right) \cdot d^\tau\right)$$

*examples from  $EX^\eta(f, \text{unif}(\{\pm 1\}^d))$  and returns a hypothesis  $h: \{\pm 1\}^d \rightarrow \{\pm 1\}$  where  $\mathbb{P}_h[d(f, h) \leq \varepsilon] \geq 1 - \delta$ .*

We use the following Fourier concentration lemma due to [17] for  $k$ -monotone Boolean functions.

► **Lemma 27** ([17]). *If  $f: \{\pm 1\}^d \rightarrow \{\pm 1\}$  is  $k$ -monotone, then  $\sum_{S: |S| > \frac{k\sqrt{d}}{\varepsilon}} \widehat{f}(S)^2 \leq \varepsilon$ .*

By Lemma 27, we can invoke Theorem 26 with  $\tau = \frac{k\sqrt{d \log N}}{\varepsilon}$ , concluding the proof of Lemma 24. ◀

### 4.3 Putting it Together: Proof of Theorem 21

**Proof.** We now have all the tools to define the algorithm and prove its correctness.

■ **Algorithm 1** Learning algorithm for  $k$ -monotone functions under product measure  $\mu$ .

---

**Input:**  $\varepsilon, \delta \in (0, 1)$  and access to examples from  $EX(f, \mu)$  where  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  is  $k$ -monotone;

1. Let  $N$  be a power of 2 such that  $\frac{8kd}{\varepsilon} \leq N \leq \frac{16kd}{\varepsilon}$ . Let  $\mathcal{A}$  denote the learning algorithm for  $k$ -monotone functions  $g: [N]^d \rightarrow \{\pm 1\}$  which has the smaller sample-complexity among the algorithms guaranteed by Lemma 23 and Lemma 24. Let  $Q$  be the sample-complexity of  $\mathcal{A}$ ;
2. Run the downsampling procedure of Proposition 22 to obtain the maps **block**, **blockpoint**, and access to the corresponding function  $f^{\text{block}}: [N]^d \rightarrow \{\pm 1\}$ ;
3. Obtain a set of  $Q$  examples  $S \in (\mathbb{R}^d \times \{\pm 1\})^Q$  from  $(EX(f, \mu))^Q$ ;
4. Let  $S^{\text{block}} = \{(\text{block}(x), f(x)): (x, f(x)) \in S\} \in ([N]^d \times \{\pm 1\})^Q$ ;
5. Run  $\mathcal{A}$  using the sample  $S^{\text{block}}$ , which returns a hypothesis  $h^{\text{block}}: [N]^d \rightarrow \{\pm 1\}$  for  $f^{\text{block}}$ ;

**Return** the hypothesis  $h: \mathbb{R}^d \rightarrow \{\pm 1\}$  for  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  defined as  $h(x) = h^{\text{block}}(\text{block}(x))$

---

Recall that given maps  $\text{block}: \mathbb{R}^d \rightarrow [N]^d$ ,  $\text{blockpoint}: [N]^d \rightarrow \mathbb{R}^d$ , and a function  $f: \mathbb{R}^d \rightarrow \{\pm 1\}$  we define the function  $f^{\text{block}}: [N]^d \rightarrow \{\pm 1\}$  as  $f^{\text{block}}(z) = f(\text{blockpoint}(z))$ . Recall that  $\text{block}(\mu)$  is the distribution over  $\text{block}(x) \in [N]^d$  when  $x \sim \mu$ . By Proposition 22, step (2) of Alg. 1 results in the following items being satisfied with probability at least  $1 - \delta$ .

1.  $\|\text{block}(\mu) - \text{unif}([N]^d)\|_{\text{TV}} \leq \frac{\delta}{Q}$ .
2.  $\mathbb{P}_{x \sim \mu} [f(x) \neq f^{\text{block}}(\text{block}(x))] \leq \varepsilon$ .

Firstly, by item (2), an example  $(\text{block}(x), f(x))$  where  $x \sim \mu$ , is equivalent to an example  $(z, b) \sim EX^\eta(f^{\text{block}}, \text{block}(\mu))$  for some  $\eta \leq \varepsilon$ . I.e. the set  $S^{\text{block}} \in ([N]^d \times \{\pm 1\})^Q$  from step (4) of Alg. 1 is distributed according to  $(EX^\eta(f^{\text{block}}, \text{block}(\mu)))^Q$ . Now, as stated, Lemma 23 and Lemma 24 only hold when  $\mathcal{A}$  is given a sample from  $(EX^\eta(f^{\text{block}}, \text{unif}([N]^d)))^Q$ . However, the following claim shows that since  $\text{block}(\mu)$  and  $\text{unif}([N]^d)$  are sufficiently close (item (1) above), the guarantees on  $\mathcal{A}$  from Lemma 23 and Lemma 24 also hold when  $\mathcal{A}$  is given a sample from  $(EX^\eta(f^{\text{block}}, \text{block}(\mu)))^Q$ .

▷ **Claim 28.** Let  $\mathcal{C}: \mathcal{X} \rightarrow \{\pm 1\}$  be a concept class and let  $\mathcal{A}$  be an algorithm which given any  $f \in \mathcal{C}$ ,  $\varepsilon, \delta \in (0, 1)$ , and  $\eta \in [0, 1/2)$  uses a sample from  $(EX^\eta(f, \text{unif}([N]^d)))^Q$  and produces  $h$  satisfying  $\mathbb{P}_{x \sim \text{unif}([N]^d)}[h(x) \neq f(x)] \leq \varepsilon$  with probability at least  $1 - \delta$ . If  $\mathcal{D}$  is a distribution over  $[N]^d$  with  $\|\mathcal{D} - \text{unif}([N]^d)\|_{TV} \leq \gamma$ , then given a sample from  $(EX^\eta(f, \mathcal{D}))^Q$ ,  $\mathcal{A}$  produces  $h$  satisfying  $\mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \leq \varepsilon + \gamma$  with probability at least  $1 - (\delta + \gamma Q)$ .

Using Claim 28 and item (1) above, if step (2) of Alg. 1 succeeds, then with probability at least  $1 - 2\delta$ , step (5) produces  $h^{\text{block}}$  such that  $\mathbb{P}_{z \sim \text{block}(\mu)}[h^{\text{block}}(z) \neq f^{\text{block}}(z)] \leq 2\varepsilon$ . By the triangle inequality and using our definition of  $h$  in the return statement of Alg. 1, we have

$$\begin{aligned} & \mathbb{P}_{x \sim \mu}[h(x) \neq f(x)] \\ & \leq \mathbb{P}_{x \sim \mu}[f(x) \neq f^{\text{block}}(\text{block}(x))] + \mathbb{P}_{x \sim \mu}[f^{\text{block}}(\text{block}(x)) \neq h^{\text{block}}(\text{block}(x))] \\ & = \mathbb{P}_{x \sim \mu}[f(x) \neq f^{\text{block}}(\text{block}(x))] + \mathbb{P}_{z \sim \text{block}(\mu)}[f^{\text{block}}(z) \neq h^{\text{block}}(z)]. \end{aligned} \quad (10)$$

The first term in the RHS is at most  $\varepsilon$  by item (2) above and the second term is at most  $2\varepsilon$  as we argued in the previous paragraph. Finally, adding up the failure probabilities of steps (2) and (5), we conclude that Alg. 1 produces  $h$  satisfying  $\mathbb{P}_{x \sim \mu}[h(x) \neq f(x)] \leq 3\varepsilon$  with probability at least  $1 - 3\delta$ . ◀

### 4.3.1 Proof of Claim 28

*Proof.* It is a well-known fact that for two distributions  $\mathcal{D}_1$  and  $\mathcal{D}_2$ , the TV-distance between the corresponding product distributions satisfies  $\|\mathcal{D}_1^Q - \mathcal{D}_2^Q\|_{TV} \leq Q \|\mathcal{D}_1 - \mathcal{D}_2\|_{TV}$  and thus we have

$$\|\mathcal{D}^Q - \text{unif}([N]^d)^Q\|_{TV} \leq \gamma Q$$

Given a set of  $Q$  examples  $S \in ([N]^d \times \{\pm 1\})^Q$ , let  $E(S)$  denote the event that the algorithm  $\mathcal{A}$  fails to produce a hypothesis with error at most  $\varepsilon$ , after sampling  $S$ . First, note the distribution over labels for the distributions are the same, and therefore

$$\begin{aligned} & \mathbb{P}_{S \sim (EX^\eta(f, \mathcal{D}))^Q}[E(S)] - \mathbb{P}_{S \sim (EX^\eta(f, \text{unif}([N]^d)))^Q}[E(S)] \\ & = \mathbb{P}_{S \sim \mathcal{D}^Q}[E(S)] - \mathbb{P}_{S \sim \text{unif}([N]^d)^Q}[E(S)]. \end{aligned} \quad (11)$$

Using the definition of TV-distance we have

$$\mathbb{P}_{S \sim \mathcal{D}^Q}[E(S)] - \mathbb{P}_{S \sim \text{unif}([N]^d)^Q}[E(S)] \leq \|\mathcal{D}^Q - \text{unif}([N]^d)^Q\|_{TV} \leq \gamma Q \quad (12)$$

and therefore

$$\mathbb{P}_{S \sim (EX^\eta(f, \mathcal{D}))^Q}[E(S)] \leq \mathbb{P}_{S \sim (EX^\eta(f, \text{unif}([N]^d)))^Q}[E(S)] + \gamma Q \leq \delta + \gamma Q \quad (13)$$

where we used  $\mathbb{P}_{S \sim (EX^\eta(f, \text{unif}([N]^d)))^\circ} [E(S)] \leq \delta$  by the assumption in the statement of the claim. Now, conditioned on  $\neg E(S)$ , we have that  $\mathcal{A}$  produces  $h$  satisfying  $\mathbb{P}_{x \sim \text{unif}([N]^d)} [h(x) \neq f(x)] \leq \varepsilon$ . Again using our bound on the TV-distance, we have

$$\mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq f(x)] - \mathbb{P}_{x \sim \text{unif}([N]^d)} [h(x) \neq f(x)] \leq \|\mathcal{D} - \text{unif}([N]^d)\|_{TV} \leq \gamma$$

and so  $\mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq f(x)] \leq \varepsilon + \gamma$ .  $\triangleleft$

## 5 Sample-Based Testing with One-Sided Error

In this section we prove Theorem 5, our upper and lower bound on sample-based testing with one-sided error over the hypercube.

**Proof of Theorem 5.** By a coupon-collecting argument, there is an  $O(d \cdot 2^d)$  sample upper bound for *exactly learning* any function over  $\{0, 1\}^d$  under the uniform distribution and therefore the upper bound is trivial.

It suffices to prove the lower bound for the case of  $r = 2$  and  $k = 1$ , i.e. for testing monotonicity of Boolean functions. We will need the following fact.

► **Fact 29.** *Let  $A \subset \{0, 1\}^d$  be any anti-chain and let  $\ell: A \rightarrow \{0, 1\}$  be any labelling of  $A$ . Then there exists a monotone function  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  such that  $f(x) = \ell(x)$  for all  $x \in A$ . I.e.  $A$  shatters the class of monotone functions.*

Now, let  $T$  be any monotonicity tester with one-sided error and let  $S \subseteq \{0, 1\}^d$  denote a set of  $s$  i.i.d. uniform samples. Since  $T$  has one-sided error, if the input function is monotone, then  $T$  must accept. In other words, for  $T$  to reject it must be sure without a doubt that the input function is not monotone. By Fact 29 for  $T$  to be sure the input function is not monotone, it must be that  $S$  is *not* an anti-chain. Let  $f: \{0, 1\}^d \rightarrow \{0, 1\}$  be any function which is  $\varepsilon$ -far from monotone. Since  $T$  is a valid tester, it rejects  $f$  with probability at least  $2/3$ . By the above argument we have

$$2/3 \leq \mathbb{P}_S [T \text{ rejects } f] \leq \mathbb{P}_S [S \text{ is not an anti-chain}] \leq s^2 \cdot \mathbb{P}_{x, y \sim \{0, 1\}^d} [x \preceq y] \quad (14)$$

where the last inequality is by a union bound over all pairs of samples. We then have

$$\mathbb{P}_{x, y \sim \{0, 1\}^d} [x \preceq y] = \mathbb{P}_{x, y \sim \{0, 1\}^d} [x_i \leq y_i, \forall i \in [d]] = \prod_{i=1}^d \mathbb{P}_{x_i, y_i \sim \{0, 1\}} [x_i \leq y_i] = (3/4)^d. \quad (15)$$

Thus, combining Equation (14) and Equation (15) yields  $s \geq \sqrt{\frac{2}{3} \left(\frac{4}{3}\right)^d} = \exp(\Omega(d))$ .  $\blacktriangleleft$

---

## References

- 1 Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Information and Computation*, 204(11):1704–1717, 2006.
- 2 Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures Algorithms*, 31(3):371–383, 2007.
- 3 Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2012. doi: 10.1109/FOCS.2012.64.
- 4 Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2016.

- 5 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. *Algorithmica*, 81(3):1247–1266, 2019. doi:10.1007/s00453-018-0467-9.
- 6 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. *Random Struct. Algorithms*, 54(3):413–443, 2019. doi:10.1002/rsa.20797.
- 7 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev.  $L_p$ -testing. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2014.
- 8 Arnab Bhattacharyya. A note on the distance to monotonicity of boolean functions. Technical Report 012, Electronic Colloquium on Computational Complexity (ECCC), 2008.
- 9 Arnab Bhattacharyya, Elena Grigorescu, Madhav Jha, Kyoming Jung, Sofya Raskhodnikova, and David Woodruff. Lower bounds for local monotonicity reconstruction from transitive-closure spanners. *SIAM Journal on Discrete Mathematics (SIDMA)*, 26(2):618–646, 2012.
- 10 Hadley Black, Eric Blais, and Nathaniel Harms. Testing and learning convex sets in the ternary hypercube. In *15th Innovations in Theoretical Computer Science Conference, ITCS, 2024*. doi:10.4230/LIPICs.ITCS.2024.15.
- 11 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A  $o(d) \cdot \text{polylog}(n)$  monotonicity tester for Boolean functions over the hypergrid  $[n]^d$ . In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018.
- 12 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Domain reduction for monotonicity testing: A  $o(d)$  tester for boolean functions in  $d$ -dimensions. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2020*. doi:10.1137/1.9781611975994.122.
- 13 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. A  $d^{1/2+o(1)}$  monotonicity tester for boolean functions on  $d$ -dimensional hypergrids. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2023*. doi:10.1109/FOCS57990.2023.00110.
- 14 Hadley Black, Deeparnab Chakrabarty, and C. Seshadhri. Directed isoperimetric theorems for boolean functions on the hypergrid and an  $\tilde{O}(n\sqrt{d})$  monotonicity tester. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC, 2023*. doi:10.1145/3564246.3585167.
- 15 Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric inequalities for real-valued functions with applications to monotonicity testing. *Random Structures & Algorithms*, 2024.
- 16 Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.
- 17 Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. In *RANDOM, 2015*. doi:10.4230/LIPICs.APPROX-RANDOM.2015.512.
- 18 Eric Blais, Renato Ferreira Pinto Jr, and Nathaniel Harms.  $\mathbb{V}_c$  dimension and distribution-free sample-based testing. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 504–517, 2021.
- 19 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings, IEEE Conference on Computational Complexity (CCC)*, 2014.
- 20 Eric Blais and Yuichi Yoshida. A characterization of constant-sample testable properties. *Random Struct. Algorithms*, 55(1):73–88, 2019. doi:10.1002/rsa.20807.
- 21 Mark Braverman, Subhash Khot, Guy Kindler, and Dor Minzer. Improved monotonicity testers via hypercube embeddings. In *14th Innovations in Theoretical Computer Science Conference, ITCS, 2023*. doi:10.4230/LIPICs.ITCS.2023.25.
- 22 Jop Briët, Sourav Chakraborty, David García Soriano, and Ari Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.
- 23 Nader H. Bshouty and Christino Tamon. On the fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996. doi:10.1145/234533.234564.

- 24 Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing  $k$ -monotonicity: The rise and fall of boolean functions. *Theory Comput.*, 15:1–55, 2019. doi:10.4086/toc.2019.v015a001.
- 25 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015.
- 26 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2013.
- 27 Deeparnab Chakrabarty and C. Seshadhri. An  $o(n)$  monotonicity tester for Boolean functions over the hypercube. *SIAM Journal on Computing (SICOMP)*, 45(2):461–472, 2014.
- 28 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014. doi:10.4086/toc.2014.v010a017.
- 29 Xi Chen, Anindya De, Yuhao Li, Shivam Nadimpalli, and Rocco A. Servedio. Mildly exponential lower bounds on tolerant testers for monotonicity, unateness, and juntas. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA, 2024*. doi:10.1137/1.9781611977912.151.
- 30 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost)  $O(n^{1/2})$  non-adaptive queries. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2015.
- 31 Xi Chen, Adam Freilich, Rocco A. Servedio, and Timothy Sun. Sample-based high-dimensional convexity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM, 2017*. doi:10.4230/LIPICS.APPROX-RANDOM.2017.37.
- 32 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2014.
- 33 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond talagrand: New lower bounds for testing monotonicity and unateness. In *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2017.
- 34 Yevgeny Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. *Proceedings, International Workshop on Randomization and Computation (RANDOM)*, 1999.
- 35 Funda Ergun, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000.
- 36 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Trans. on Algorithms (TALG)*, 6(3), 2010.
- 37 Renato Ferreira Pinto Jr and Nathaniel Harms. Distribution testing under the parity trace, 2023. arXiv:2304.01374.
- 38 Renato Ferreira Pinto Jr and Nathaniel Harms. Distribution testing with a confused collector. In *15th Innovations in Theoretical Computer Science Conference, ITCS, 2024*. doi:10.4230/LIPICS.ITCS.2024.47.
- 39 Eldar Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.
- 40 Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *Innovations in Theoretical Computer Science, ITCS*. ACM, 2014. doi:10.1145/2554797.2554841.
- 41 Eldar Fischer, Oded Lachish, and Yadu Vasudev. Trading query complexity for sample-based testing and multi-testing scalability. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.75.



- 42 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, and Ronitt Rubinfeld. Monotonicity testing over general poset domains. *Proceedings, ACM Symposium on Theory of Computing (STOC)*, 2002.
- 43 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.
- 44 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- 45 Oded Goldreich and Dana Ron. On sample-based testers. *ACM Trans. Comput. Theory*, 8(2):7:1–7:54, 2016. doi:10.1145/2898355.
- 46 Elena Grigorescu, Akash Kumar, and Karl Wimmer. Flipping out with many flips: Hardness of testing k-monotonicity. *SIAM J. Discret. Math.*, 33(4):2111–2125, 2019. doi:10.1137/18M1217978.
- 47 Shirley Halevy and Eyal Kushilevitz. Distribution-free property testing. *Proceedings, International Workshop on Randomization and Computation (RANDOM)*, 2003.
- 48 Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Structures Algorithms*, 33(1):44–67, 2008.
- 49 Nathaniel Harms and Yuichi Yoshida. Downsampling for testing and learning in product distributions. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022*, 2022. doi:10.4230/LIPIcs.ICALP.2022.71.
- 50 Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.
- 51 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000. doi:10.1006/jcss.1999.1656.
- 52 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and Boolean isoperimetric type theorems. In *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2015.
- 53 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM J. Comput.*, 47(6):2238–2276, 2018. doi:10.1137/16M1065872.
- 54 Jane Lange, Ronitt Rubinfeld, and Arsen Vasilyan. Properly learning monotone functions via local correction. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2022. doi:10.1109/FOCS54457.2022.00015.
- 55 Jane Lange and Arsen Vasilyan. Agnostic proper learning of monotone functions: beyond the black-box correction barrier. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, 2023. doi:10.1109/FOCS57990.2023.00068.
- 56 Eric Lehman and Dana Ron. On disjoint chains of subsets. *Journal of Combinatorial Theory, Series A*, 94(2):399–404, 2001.
- 57 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. doi:10.1145/174130.174138.
- 58 Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Nithin Varma. Parameterized property testing of functions. *ACM Trans. Comput. Theory*, 9(4):17:1–17:19, 2018. doi:10.1145/3155296.
- 59 Sofya Raskhodnikova. Monotonicity testing. *Masters Thesis, MIT*, 1999.
- 60 Dana Ron, Ronitt Rubinfeld, Muli Safra, and Omri Weinstein. Approximating the Influence of Monotone Boolean Functions in  $O(\sqrt{n})$  Query Complexity. In *Proceedings, International Workshop on Randomization and Computation (RANDOM)*, 2011.
- 61 R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal of Computing*, 25:647–668, 1996.
- 62 Michael E. Saks and C. Seshadhri. Parallel monotonicity reconstruction. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.
- 63 Michel Talagrand. How much are increasing sets positively correlated? *Comb.*, 16(2):243–258, 1996. doi:10.1007/BF01844850.



# Approximating the Number of Relevant Variables in a Parity Implies Proper Learning

Nader H. Bshouty  

Department of Computer Science, Technion, Israel

George Haddad 

Department of Computer Science, Technion, Israel

---

## Abstract

Consider the model where we can access a parity function through random uniform labeled examples in the presence of random classification noise. In this paper, we show that approximating the number of relevant variables in the parity function is as hard as properly learning parities.

More specifically, let  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ , where  $\gamma(x) \geq x$ , be any strictly increasing function. In our first result, we show that from any polynomial-time algorithm that returns a  $\gamma$ -approximation,  $D$  (i.e.,  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ ), of the number of relevant variables  $d(f)$  for any parity  $f$ , we can, in polynomial time, construct a solution to the long-standing open problem of polynomial-time learning  $k(n)$ -sparse parities (parities with  $k(n) \leq n$  relevant variables), where  $k(n) = \omega_n(1)$ .

In our second result, we show that from any  $T(n)$ -time algorithm that, for any parity  $f$ , returns a  $\gamma$ -approximation of the number of relevant variables  $d(f)$  of  $f$ , we can, in polynomial time, construct a  $\text{poly}(\Gamma(n))T(\Gamma(n)^2)$ -time algorithm that properly learns parities, where  $\Gamma(x) = \gamma(\gamma(x))$ .

If  $T(\Gamma(n)^2) = \exp(o(n/\log n))$ , this would resolve another long-standing open problem of properly learning parities in the presence of random classification noise in time  $\exp(o(n/\log n))$ .

**2012 ACM Subject Classification** Theory of computation

**Keywords and phrases** PAC Learning, Random Classification Noise, Uniform Distribution, Parity, Sparsity Approximation

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.38

**Category** RANDOM

**Acknowledgements** We would like to thank the anonymous reviewer of RANDOM for providing another approach for finding the relevant variables in the target function. We also extend our gratitude to the other reviewers for their useful comments and suggestions, which have greatly improved this manuscript.

## 1 Introduction

The problem of PAC learning parity, with and without noise, and approximating its sparsity has been extensively studied in the literature. See [2, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 32, 33, 35] and references therein.

In properly learning parities under the uniform distribution, the learner can observe labeled examples  $\{(a_i, b_i)\}_i$ , where  $b_i = f(a_i)$ ,  $a_i$  are drawn independently from the uniform distribution, and  $f$  is the target parity. The goal is to return the target parity function  $f$  exactly.

In the random classification noise model with noise rate  $\eta$ , [1], each label  $b_i$  is independently flipped (misclassified) with probability  $\eta$ . The problem of learning parities with noise (LPN) is known to be computationally challenging. Some evidence of its hardness comes from the fact that it cannot be learned efficiently in the so-called statistical query (SQ) model [27] under the uniform distribution [9, 12]. LPN serves as the foundation for several cryptographic constructions, largely because its hardness in the presence of noise is assumed. See for example [10, 31].



© Nader H. Bshouty and George Haddad;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 38; pp. 38:1–38:15



Leibniz International Proceedings in Informatics

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

While PAC learning of parities (and thus determining its sparsity) under the uniform distribution can be accomplished in polynomial time using Gaussian elimination, addressing this problem in the presence of random classification noise remains one of the most long-standing challenges in learning theory. The only known algorithm is that of Blum et al. [11], which runs in time  $2^{O(n/\log n)}$ , requires  $2^{O(n/\log n)}$  labeled examples, and handles only a constant noise rate. This algorithm holds the record as the best-known solution for this problem. Finding a  $2^{o(n/\log n)}$ -time learning algorithm for parities or proving the impossibility of such an algorithm remains a significant and unresolved challenge.

When the number of relevant variables<sup>1</sup>  $k$  of the parity function  $f$  is known ( $f$  is called  $k$ -sparse parity), all the algorithms proposed in the literature run in time  $n^{ck}$  for some constant  $c < 1$ , [5, 7, 22, 33, 36]. Finding a polynomial-time algorithm for  $k$ -sparse parities for some  $k = \omega(1)$ , or proving the impossibility of such an algorithm, is another significant and unresolved challenge.

In a related vein, another challenging problem is determining or approximating the sparsity of the parity function, i.e., the number of relevant variables in the target function. This problem was studied in the PAC-learning model [34] under specific<sup>2</sup> distributions [2, 3, 4, 7, 16, 17, 18, 19, 20, 30, 35].

For the problem of determining the sparsity under any distribution and without noise, Downey et al. [18] and Bhattacharyya et al. [4] show that determining the sparsity  $k$  of parities is  $W[1]$ -hard. Bhattacharyya et al. [7] show that the time complexity is  $\min(2^{\Theta(n)}, n^{\Theta(k)})$ , assuming 3-SAT has no  $2^{o(n)}$ -time algorithm. For the problem of approximating the sparsity, Dumer et al. [19] showed that if  $\text{RP} \neq \text{NP}$ , then it is hard to approximate the sparsity within some constant factor  $\gamma > 1$ . See also [2, 16, 17, 30]. When the distribution is uniform, there is a polynomial-time algorithm that uses  $O(n)$  labeled examples and learns parities using Gaussian elimination, thereby determining their sparsity.

In this paper, we pose the question: Can we approximate the sparsity of the parity function in polynomial time using random uniform labeled examples in the presence of random classification noise? We show that approximating the number of relevant variables in the parity function is as hard as properly learning parities.

We first show the following.

► **Theorem 1.** *Let  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be any strictly increasing function, where  $\gamma(x) \geq x$ . Consider a polynomial-time algorithm that, for any parity  $f$ , uses random uniform labeled examples of  $f$  in the presence of random classification noise and returns an integer  $D$  such that<sup>3</sup>  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ , where  $d(f)$  is the number of relevant variables in  $f$ . One can, in polynomial time, construct an algorithm that runs in polynomial time, uses random uniform labeled examples in the presence of random classification noise, and learns  $k(n)$ -sparse parities for some<sup>4</sup>  $k(n) = \omega_n(1)$ .*

This would solve the long-standing open problem of polynomial-time learning  $k$ -sparse parities for some  $k = \omega_n(1)$ .

We then show that

<sup>1</sup> A variable is *relevant* in  $f$  if  $f$  depends on that variable.

<sup>2</sup> Some of the problems are introduced as follows: Given a matrix  $M \in F_2^{m \times n}$ , a vector  $b \in \{0, 1\}^m$ , and an integer  $k$ . Deciding if there exists a weight  $k$  vector  $x \in \{0, 1\}^n$  such that  $Mx = b$ . This is equivalent to the decision problem when the distribution is uniform over the rows of  $M$ .

<sup>3</sup> See Section 1.4 for the justification of why we use this definition and not the standard definition  $d(f) \leq D \leq \gamma(d(f))$ .

<sup>4</sup> Throughout this paper, we also have  $k = n - \omega(1)$ . For  $k = O(1)$  and  $k = n - O(1)$ , there are polynomial-time learning algorithms.

► **Theorem 2.** *From any  $T(n)$ -time algorithm that, for any parity  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , uses  $Q(n)$  random uniform labeled examples of  $f$  in the presence of random classification noise and returns a  $\gamma$ -approximation of the number of relevant variables  $d(f)$  of  $f$ , one can, in polynomial time, construct a  $\text{poly}(\Gamma(n))T(\Gamma(n)^2)$ -time algorithm that uses  $\text{poly}(\Gamma(n))Q(\Gamma(n)^2)$  random uniform labeled examples in the presence of random classification noise and properly learns parities, where  $\Gamma(x) = \gamma(\gamma(x))$ .*

If  $T(\Gamma(n)^2) = \exp(o(n/\log n))$ , this would resolve another long-standing open problem of proper learning parities in the presence of random classification noise in time  $\exp(o(n/\log n))$ . This is applicable, for example, for any  $\text{poly}(\cdot)$ -approximation and  $\exp(n^{1/c})$ -time algorithm for some sufficiently large constant  $c$ . As well as to quasi- $\text{poly}(\cdot)$ -approximation and  $\exp(\exp(\log n)^{1/c})$ -time algorithm for some sufficiently large constant  $c$ .

In this paper, while the above discussions and the technique section have been primarily focused on parities, that is, linear functions over the binary field  $\mathbb{F}_2$ , the results we present in this paper are not limited to this specific case. We generalize our result to encompass any linear function over any finite field. This extension allows our results to be applicable to a broader range of linear systems beyond the binary paradigm, effectively widening their relevance in coding theory and cryptography.

## 1.1 Our Technique

In this section, we present the technique used in the paper to prove the results in Theorem 1 and 2.

For learning in the presence of random classification noise, when the noise rate  $\eta = 1/2$ , the labels will be randomly uniform, and learning is impossible. Therefore, we must assume that the learner knows some upper bound  $\eta_b < 1/2$  for  $\eta$  [1].

## 1.2 Approximation Implies Learning $k$ -Sparse Parities

In this section, we present two approaches that prove Theorem 1. The first is our method, and the second was suggested by an anonymous reviewer of RANDOM.

While the approach suggested by the anonymous reviewer is truly inspiring, we believe that our method offers significant value, is worth presenting in this paper, and may be useful for solving other problems.

### 1.2.1 First Approach

In this section, we will outline the technique for the binary field, though some essential details are omitted to provide a broader overview of the main concepts and approach. Additionally, proving the result for any field requires more careful treatment.

Let  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be any strictly increasing function such that for every<sup>5</sup>  $x > 1$ ,  $\gamma(x) > x$ .

Let  $\mathcal{A}$  be a polynomial-time randomized algorithm that  $\gamma$ -approximates the number of relevant variables  $d(f)$  in a parity  $f$ , using random uniform labeled examples of  $f$  in the presence of random classification noise with any noise rate<sup>6</sup>  $\eta \leq \eta_b$ . Thus, for every parity  $f$  with  $d(f)$  relevant variables, with probability at least  $1 - \delta$  we have  $\gamma^{-1}(d(f)) \leq \mathcal{A}(f) \leq \gamma(d(f))$ . We will demonstrate how to construct a polynomial-time learning algorithm for  $k(n)$ -sparse parities, for some  $k(n) = \omega_n(1)$ . First, we will show how to find  $k(n)$ .

<sup>5</sup> We need this constraint to ensure that  $\gamma^{-1}(x) < \gamma(x)$  for every  $x > 1$ .

<sup>6</sup> Here,  $\eta$  is not known to the algorithm, but  $\eta_b$  is known.

## 38:4 Approximating the Number of Relevant Variables in a Parity Implies Proper Learning

Let  $\text{Lin}(d)$  be the class of  $d$ -sparse parities. Assume for now that the noise rate is  $\eta_b$ . Later, we will show how to modify the algorithm to work for any unknown noise rate  $\eta \leq \eta_b$ .

We first use the algorithm  $\mathcal{A}$  to construct a table that provides values which approximate

$$\Psi_{\mathcal{A}}(d) = \mathbf{E}_{(f,s) \sim_u \text{Lin}(d) \times S(f)} [\mathcal{A}(f)]$$

with additive error of  $1/\text{poly}(n)$ , for every  $d \in [n]$  and noise rate  $\eta_b$ . Here  $f$  is a  $d$ -sparse parity chosen uniformly at random from  $\text{Lin}(d)$ , and  $s$  is a uniformly random string in  $S(f)$  - the set of random bits used by the algorithm (for the randomness of the algorithm and the noise) for which the algorithm returns a correct answer, namely, returns  $D$  such that  $\gamma^{-1}(d) \leq D \leq \gamma(d)$ .

To approximate  $\Psi_{\mathcal{A}}(d)$  for some  $d \in [n]$ , we iterate a polynomial number of times. At each iteration, we draw a random uniform  $f \in \text{Lin}(d)$  and run  $\mathcal{A}$ . For each labeled example requested by  $\mathcal{A}$ , we draw a random uniform  $u \in \{0,1\}^n$  and compute  $v = f(u)$ . We then, with probability  $\eta_b$ , return<sup>7</sup>  $(u, v + 1)$  to  $\mathcal{A}$ , and, with probability  $1 - \eta_b$ , return  $(u, v)$ . If the algorithm outputs an integer  $D$  such that  $\gamma^{-1}(d) \leq D \leq \gamma(d)$ , we retain that  $D$ . Otherwise, we repeat the process. Obviously,  $\mathbf{E}[D] = \Psi_{\mathcal{A}}(d)$ , and therefore, using Hoeffding's bound, such a table can be constructed in polynomial time.

Now, using the fact that  $\gamma$  is strictly increasing and  $\gamma^{-1}(d) \leq \Psi_{\mathcal{A}}(d) \leq \gamma(d)$ , and applying a basic averaging argument, we show that there exists a  $k := k(n) = \omega_n(1)$  for which  $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k-1) \geq 1/\text{poly}(n)$ . We now show how to learn  $k$ -sparse parities with noise rate  $\eta_b$  in polynomial time and afterward for any  $\eta \leq \eta_b$ .

Suppose that the target function  $f \in \text{Lin}(k)$  can be accessed through random uniform labeled examples in the presence of random classification noise with noise rate  $\eta_b$ . We first show how to approximate  $\Psi_{\mathcal{A}}(d(f(x) + x_i))$  for any  $i \in [n]$  without knowing  $f$ . Recall that  $d(f(x) + x_i)$  is the number of relevant variables in  $f(x) + x_i$ . The key idea here is that if  $(a, b)$  is a labeled example of  $f$ , then for a random uniform permutation  $\phi$ ,  $((a_{\phi^{-1}(1)}, \dots, a_{\phi^{-1}(n)}), b + a_i)$  is a labeled example of the function  $f(x_{\phi(1)}, \dots, x_{\phi(n)}) + x_{\phi(i)}$  which is a random and uniform drawn function in  $\text{Lin}(d(f(x) + x_i))$ . Therefore, using Hoeffding's Bound, we can approximate  $\Psi_{\mathcal{A}}(d(f(x) + x_i))$  for every  $i \in [n]$ .

Now,  $x_i$  is relevant in  $f(x)$  if and only if  $f(x) + x_i \in \text{Lin}(k-1)$  and then  $\Psi_{\mathcal{A}}(d(f(x) + x_i)) = \Psi_{\mathcal{A}}(k-1)$ . On the other hand,  $x_i$  is not relevant in  $f(x)$  if and only if  $f(x) + x_i \in \text{Lin}(k+1)$ , and then  $\Psi_{\mathcal{A}}(d(f(x) + x_i)) = \Psi_{\mathcal{A}}(k+1)$ . Since  $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k-1) \geq 1/\text{poly}(n)$ , these two cases are distinguishable in polynomial time. Consequently, we can differentiate between variables in  $f$  that are relevant and those that are not. This gives the learning algorithm to  $\text{Lin}(k)$  when the noise rate is  $\eta_b$ .

This algorithm runs in time  $T = \text{poly}(n, 1/(1 - 2\eta_b))$ . When  $\eta$  is not known, we can run the above procedure for all possible values  $\eta^{(j)} = 1/2 - j/T^c$ , where  $c$  is a sufficiently large constant, and  $j \in [T^c/2] \cup \{1\}$ . For each  $j$ , when the algorithm receives a labeled example  $(u, v)$ , we magnify the error rate to  $\eta_b$  by drawing  $\xi \in \{0,1\}$ , which is equal to 1 with probability  $(\eta_b - \eta^{(j)})/(1 - 2\eta^{(j)})$ , and returning  $(u, v + \xi)$  to the algorithm. This new labeled example has noise rate  $\eta_b$ . We collect all the  $T^c/2 + 1$  hypotheses generated from the outputs and then employ a standard algorithm to select the one closest to the target [1]. The result follows because there exists a  $j$  such that<sup>8</sup>  $|\eta^{(j)} - \eta| \leq 1/T^c$ . Consequently, using the total variation distance, one of the hypotheses is the target.

<sup>7</sup> Here, + is exclusive or.

<sup>8</sup> If  $\eta_j = \eta + \epsilon$  then the magnified noise is  $\eta_b + \lambda\epsilon$  where  $\lambda = (\eta + \eta_b - 1 + \epsilon)/(1 - 2(\eta + \epsilon))$ .

In this paper, we extend our result to any linear function over any finite field  $\mathbb{F}$ . The approach used is similar to the case of parities (the binary field  $\mathbb{F} = \{0, 1\}$ ) with some technical but nontrivial modifications.

### 1.2.2 The Second Approach

This second approach was suggested by an anonymous reviewer of RANDOM, whose insightful comments and suggestions significantly improved this manuscript for the case of the binary field. For non-binary fields, this approach can identify the relevant variables of the function. We then use the approach developed in Lemma 9 and Lemma 10 to find the coefficients of the relevant variables.

Suppose there exists a randomized algorithm  $\mathcal{A}(n)$  that runs in time  $T(n)$  and  $\gamma$ -approximates the number of relevant variables in a parity function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , using random uniform labeled examples of  $f$  in the presence of random classification noise with a noise rate  $\eta \leq \eta_b$ . Here, too, the algorithm needs to know an upper bound on  $\eta$ . We will explain the reasons for this below.

Let  $k_1 = \omega_n(1)$  be an integer such that  $k_2 = \gamma(\gamma(k_1) + 1) = n - \omega_n(1)$ . For any  $k_1$ -sparse parity  $f$ , the algorithm outputs  $\mathcal{A}(f) \in [\gamma^{-1}(k_1), \gamma(k_1)]$ , and for any  $k_2$ -sparse parity function  $g$ , it outputs  $\mathcal{A}(g) \in [\gamma(k_1) + 1, \gamma(\gamma(k_1) + 1)]$ . Since the two intervals are disjoint, the algorithm can distinguish between  $k_1$ -sparse parities and  $k_2$ -sparse parities in polynomial time. Let  $\mathcal{B}$  be the polynomial-time algorithm that distinguishes between them.

Consider the algorithm  $\mathcal{B}$  when it runs on random uniform examples with random uniform labels. Suppose that with probability  $p$ , the algorithm answers that the function is a  $k_1$ -sparse parity, and with probability  $1 - p$ , it answers that it is a  $k_2$ -sparse parity. If  $p > 1/2$ , then with probability at least  $1/2$ ,  $\mathcal{B}$  can distinguish between  $k_2$ -sparse parities and random uniform examples with random uniform labels. Otherwise, with probability at least  $1/2$ ,  $\mathcal{B}$  can distinguish between  $k_1$ -sparse parities and random uniform examples with random uniform labels.

Suppose, without loss of generality, the latter holds. We now give an algorithm that finds the relevant variables when the target function is a  $k_1$ -parity function and, consequently, learns  $k_1$ -sparse parities. This algorithm is from [10].

For every  $i \in [n]$ , we run the algorithm  $\mathcal{B}$  and change the  $i$ -th coordinate of each example to a random uniform element in  $\{0, 1\}$ . If  $x_i$  is not a relevant variable of  $f$ , then the labeled examples are labeled examples of  $f$ , and the algorithm answers that it is a  $k_1$ -parity function. If  $x_i$  is a relevant variable of  $f$ , then it is easy to see that the new labeled examples are random uniform with random uniform labels, and the algorithm answers accordingly. This distinguishes between variables in  $f$  from those that are not in  $f$ .

In this method, too, we must know some upper bound on  $\eta$ . Otherwise, algorithms  $\mathcal{A}$  and  $\mathcal{B}$  would need to be Las Vegas algorithms, and we would not know when to stop the algorithm when dealing with random uniform examples with random uniform labels.

For the problem of finding the relevant variables of the target in other fields, the generalization of this to any field is straightforward.

## 1.3 Approximation Implies Learning Parities

In this section, we show how  $\gamma$ -approximation implies proper learning parities.

Let  $\gamma(x)$  be any strictly increasing function. Suppose there exists a randomized algorithm  $\mathcal{A}(n)$  that runs in time  $T(n)$  and  $\gamma$ -approximates the number of relevant variables in a parity  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , using random uniform labeled examples of  $f$  in the presence of random classification noise.

Let  $\Gamma(x) = \gamma(\gamma(x))$ . As in Section 1.2.1, using a basic averaging argument, we show that there exists a sequence of integers  $k_1 < k_2 < \dots < k_t < \Gamma^{-1}(n)$ , where for each  $i$ ,  $k_{i+1} < \Gamma(k_i)$  and  $\Psi_{\mathcal{A}}(k_i + 1) - \Psi_{\mathcal{A}}(k_i - 1) > 1/\text{poly}(n)$ . As before, we obtain algorithms that learn  $\text{Lin}(k_i)$  for each  $i$ .

Now, we show how to obtain a learning algorithm for  $d$ -sparse parities for every  $d < \Gamma^{-1}(\sqrt{n})$ . Given any  $d < \Gamma^{-1}(\sqrt{n})$ , there exists a  $j$  such that  $k_{j-1} < d \leq k_j$  where  $k_0 = 0$ . To learn  $d$ -sparse parities, we uniformly at random choose distinct  $i_1, \dots, i_{k_j-d} \in [n]$ , run the algorithm for learning  $k_j$ -sparse parities and modify each labeled example  $(a, b)$  to  $(a, b + a_{i_1} + \dots + a_{i_{k_j-d}})$ . If  $g(x) = f(x) + x_{i_1} + \dots + x_{i_{k_j-d}}$  is in  $\text{Lin}(k_j)$ , then the algorithm w.h.p learns  $g(x)$ . We then show that because  $d < \Gamma^{-1}(\sqrt{n})$ , with high probability, the variables  $x_{i_1}, \dots, x_{i_{k_j-d}}$  are not relevant variables in  $f$ . We can then conclude that, w.h.p,  $g \in \text{Lin}(k_j)$ . Therefore, w.h.p., we can learn  $g(x)$ , and consequently, we can learn  $f(x) = g(x) + x_{i_1} + \dots + x_{i_{k_j-d}}$ .

This provides a learning algorithm for  $d$ -sparse parities for any  $d \leq \Gamma^{-1}(\sqrt{n})$ . Recognizing that this applies to every  $n$ , we can regard  $f$  as a function over  $N := \Gamma(n)^2$  variables by adding  $\Gamma(n)^2 - n$  dummy variables and appending  $\Gamma(n)^2 - n$  random uniform elements from  $\{0, 1\}$  to each  $a$  in the labeled example  $(a, b)$ . By applying this construction to the algorithm  $\mathcal{A}(N)$ , we obtain a learning algorithm for  $d$ -sparse parity for any  $d \leq \Gamma^{-1}(\sqrt{N}) = n$ .

Now, the algorithm for learning parities can run all the learning algorithms for  $d$ -sparse parities for all  $d \leq n$ . It takes all the outputs and then employs a standard algorithm to select the one closest to the target [1]. See also Lemma 3.

### 1.4 Justification for the Use of the $\gamma$ -Approximation Definition

In our approach, we define a  $\gamma$ -approximation of the number of relevant variables  $d(f)$  in a parity function  $f$  such that  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ , instead of using the standard definition  $d(f) \leq D \leq \gamma(d(f))$ .

The key reason for this choice is that the latter definition loses its effectiveness when  $d(f)$  approaches  $n$ , the number of variables. Specifically, if  $d(f)$  is close to  $n$ , say  $O(n)$ , the condition  $d(f) \leq D \leq \gamma(d(f))$ , for  $\gamma(n) = \omega(n)$ , effectively reduces to  $d(f) \leq D \leq n$ . In this scenario, the value of  $\gamma$  becomes less significant because the approximation  $D$  would naturally fall within the trivial range of  $d(f) = O(n)$  to  $n$ .

On the other hand, our chosen definition  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$  ensures that the approximation  $D$  always depends on the function  $\gamma$ . This definition retains its utility even when  $d(f)$  is large, as  $\gamma^{-1}(d(f))$  provides a lower bound that is influenced by  $\gamma$ , thereby maintaining the approximation's relevance and precision.

Thus, by using the  $\gamma$ -approximation definition  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ , we ensure a meaningful and consistent approximation of the number of relevant variables  $d(f)$  across the entire range of possible values, preserving the value and impact of the function  $\gamma$ .

## 2 Definitions and Preliminaries

Let  $\mathbb{F}$  be any finite field and  $\mathbb{F}_q$  be the field with  $q$  elements. We define  $\text{Lin}(\mathbb{F})$  as the class of all linear functions over the field  $\mathbb{F}$ , i.e., functions  $a \cdot x$  where  $a \in \mathbb{F}^n$  and  $x = (x_1, \dots, x_n)$ . A  $d$ -sparse linear function over  $\mathbb{F}$  is a function in  $\text{Lin}(\mathbb{F})$  with  $d$  relevant variables. The class  $\text{Lin}(\mathbb{F}, d)$  is the class of all  $d$ -sparse linear functions over  $\mathbb{F}$ . When  $\mathbb{F}$  is the binary field  $\mathbb{F}_2 = \{0, 1\}$ , the functions in  $\text{Lin}(\mathbb{F}_2)$  are called parity functions, and the functions in  $\text{Lin}(\mathbb{F}_2, d)$  are called  $d$ -sparse parities. We use the notation  $\text{Lin}_n(\mathbb{F})$  and  $\text{Lin}_n(\mathbb{F}, d)$  to emphasize the number of variables.



For  $f \in \text{Lin}(\mathbb{F})$ , we denote by  $d(f)$  the number of variables on which  $f$  depends. For a strictly increasing function  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  such that  $\gamma(x) > x$  for every  $x$ , we say that an algorithm  $\mathcal{A}$   $\gamma$ -approximates  $d(f)$  in time  $T = T(n)$  and  $Q = Q(n)$  labeled examples if the algorithm runs in time  $T$ , uses  $Q$  labeled examples to  $f$ , and with probability at least  $2/3$ , returns an integer  $D$  such that  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ .

In proper learning  $\text{Lin}(\mathbb{F})$  under the uniform distribution, the learner can observe labeled examples  $(a, b)$  where  $b = f(a)$  and  $a \in \mathbb{F}^n$  are drawn independently and uniformly distributed over  $\mathbb{F}^n$ , with  $f \in \text{Lin}(\mathbb{F})$  being the target linear function. The goal is to (properly) exactly return the linear function  $f$ . In the random classification noise model with noise rate  $\eta$ , each label  $b$  is equal to  $f(a)$  with probability  $1 - \eta$  and is a random uniform element in  $\mathbb{F} \setminus \{f(a)\}$  with probability  $\eta$ .

When  $\eta = 1 - 1/|\mathbb{F}|$ , the label is a random uniform element of  $\mathbb{F}$ ; hence, learning is impossible. Therefore, we must assume that the learner knows some upper bound  $\eta_b < 1 - 1/|\mathbb{F}|$  for  $\eta$  [1]. When  $\eta = \eta_b$ , to distinguish between labeled examples with random uniform labels and the function  $f(x) = 0$ , we need at least  $1/(1 - \eta_b - 1/|\mathbb{F}|)$  labeled examples. Therefore, a polynomial-time algorithm in this model is an algorithm that runs in time  $\text{poly}(1/(1 - \eta_b - 1/|\mathbb{F}|), n, 1/\delta)$  [1].

The following Lemma shows how to learn when the algorithm has unlimited computational power.

► **Lemma 3.** *Let  $C \subseteq \text{Lin}(\mathbb{F})$ . Then  $C$  is learnable under the uniform distribution in the random classification noise model in time  $\tilde{O}(|C| \log(1/\delta)/(1 - \eta_b - 1/|\mathbb{F}|)^2)$  from*

$$Q = \frac{\log \frac{|C|}{\delta}}{(1 - \eta_b - 1/|\mathbb{F}|)^2}$$

*labeled examples.*

**Proof.** Let  $(a, b)$  be a labeled example and  $f$  be the target function. Then

$$\Pr[f(a) = b] = \eta \Pr[f(a) = b | b \neq f(a)] + (1 - \eta) \Pr[f(a) = b | b = f(a)] = 1 - \eta \geq 1 - \eta_b.$$

If  $g \neq f$  and  $g \in \text{Lin}(\mathbb{F})$  then

$$\Pr[g(a) = b] = \eta \Pr[g(a) = b | b \neq f(a)] + (1 - \eta) \Pr[g(a) = b | b = f(a)] = \frac{1}{|\mathbb{F}|}.$$

The result now follows by applying Chernoff's bound to estimate  $\Pr[g(a) = b]$  for all  $g \in C$  with confidence of  $1 - \delta/|C|$  and an additive error of  $(1 - \eta_b - 1/|\mathbb{F}|)/4$ . ◀

The following lemma shows that, in approximation algorithms, the dependency on  $\delta$  is logarithmic. This is a well-known result. For completeness, a sketch of the proof is provided.

► **Lemma 4.** *If there exists an algorithm  $\mathcal{A}$  that runs in time  $T(n)$ , uses  $Q(n)$  labeled examples to  $f \in \text{Lin}(\mathbb{F}, d)$  according to the uniform distribution in the presence of random classification noise and, with probability at least  $2/3$ , returns a  $\gamma$ -approximation of  $d(f)$ , then there is an algorithm that runs in time  $O(T(n) \log(1/\delta))$ , uses  $O(Q(n) \log(1/\delta))$  labeled examples to  $f \in \text{Lin}(\mathbb{F}, d)$  according to the uniform distribution in the presence of random classification noise, and with probability at least  $1 - \delta$ , returns a  $\gamma$ -approximation of  $d(f)$ .*

**Proof.** We run  $\mathcal{A}$ ,  $O(\log(1/\delta))$  times and take the median of the outputs. The correctness of this algorithm follows from an application of Chernoff's bound. ◀

The same is true for learning.

► **Lemma 5.** *If there exists an algorithm  $\mathcal{A}$  that runs in time  $T(n)$ , uses  $Q(n)$  labeled examples to  $f \in \text{Lin}(\mathbb{F}, d)$  according to the uniform distribution in the presence of random classification noise and, with probability at least  $2/3$ , properly learns the target  $f$ , then there is an algorithm that runs in time  $O(T(n) \log(1/\delta))$ , uses  $O(Q(n) \log(1/\delta))$  labeled examples to  $f \in \text{Lin}(\mathbb{F}, d)$  according to the uniform distribution in the presence of random classification noise, and with probability at least  $1 - \delta$ , learns the target  $f$ .*

**Proof.** Since  $\mathcal{A}$  properly learns  $f$ , we run the algorithm  $O(\log(1/\delta))$  times and output the hypothesis that occurs most frequently in the output. ◀

### 3 Approximation vs. Learning

In this section, we prove the two results.

#### 3.1 Approximation Implies Learning Some $\text{Lin}(\mathbb{F}, k)$

In this section, we prove that approximating the number of relevant variables in the parity function implies polynomial-time properly learning  $\text{Lin}(\mathbb{F}, k(n))$  for some  $k(n) = \omega_n(1)$ .

We prove.

► **Theorem 6.** *Let  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be any strictly increasing function where  $\gamma(x) > x$  for every  $x$ . Let  $\pi(n)$  be any function such that  $\pi(n) = \omega_n(1)$ . Consider any polynomial-time algorithm  $\mathcal{A}'(n)$  that, for any linear function  $f \in \text{Lin}(\mathbb{F})$ , uses random uniform labeled examples of  $f$  in the presence of random classification noise and, with probability at least  $2/3$ , returns a  $\gamma$ -approximation of the number of relevant variables  $d(f)$  of  $f$ . From  $\mathcal{A}'(n)$ , one can, in polynomial time, construct a  $\text{poly}(n, 1/(1 - \eta_b - 1/|\mathbb{F}|), \min(|\mathbb{F}|, 1/(1 - \eta_b)^{\pi(n)}))$ -time algorithm that properly learns  $\text{Lin}(\mathbb{F}, k(n))$  from a polynomial number of random uniform labeled examples in the presence of random classification noise for some  $k(n) = \omega_n(1)$ .*

We will assume for now that the noise rate  $\eta = \eta_b$  is known. In Section 1.2.1, we showed how to handle unknown noise rates  $\eta \leq \eta_b$ . Recall that a polynomial-time algorithm in this model is an algorithm that runs in time  $\text{poly}(1/(1 - \eta_b - 1/|\mathbb{F}|), n, 1/\delta)$ , [1]. In particular, the algorithm constructed in Theorem 6 runs in polynomial time for either

- Any  $\eta_b$  and fields of size<sup>9</sup>  $|\mathbb{F}| = \text{poly}(n)$ , or
- Any field  $\mathbb{F}$  when  $\eta_b \leq 1 - 1/|\mathbb{F}| - 1/\psi(n)$ , where  $\psi(n) = 2^{o(\log(n))}$ .

Let  $\mathcal{A}'(n, s, f)$  be any algorithm that uses random uniform labeled examples of  $f \in \text{Lin}_n(\mathbb{F})$  in the presence of random classification noise and, with probability at least  $2/3$ , returns a  $\gamma$ -approximation of the number of relevant variables,  $d(f)$ , of  $f$ . The new parameter  $s$  is added for the random bits used in the algorithm for its coin flips and the noise. First, we will use Lemma 4 to make the algorithm's success probability  $1 - \delta'$  for a fixed, sufficient small  $\delta'$  that depends on  $n$  and  $|\mathbb{F}|$ . For the proof of the Theorem in this section,  $\delta' = 1/(|\mathbb{F}|n^7)$  suffices. By Lemma 4, this adds a factor of  $O(\log n + \log |\mathbb{F}|)$  to the time and the number of labeled examples which will be swallowed by the  $\tilde{O}(\cdot)$  in the final time and sample complexity. Second, we will modify the output of the algorithm to  $\min(\gamma(D_f), n)$ , where  $D_f$  is the output of the latter algorithm. Let the resulting algorithm be denoted as  $\mathcal{A}$ . We will denote the algorithm's output by  $\mathcal{A}(n, s, f)$ . Consequently, we will have that, with probability at least  $1 - \delta'$ ,

$$d(f) \leq \mathcal{A}(n, s, f) \leq \Delta(d(f)) \leq n \tag{1}$$

<sup>9</sup> This makes sense when we have a sequence of fields  $\mathbb{F}_i$  such that  $\mathbb{F}_i \subseteq \mathbb{F}_{i+1}$  and  $|\mathbb{F}_n| = \text{poly}(n)$ .

where

$$\Delta(x) = \min(\gamma(\gamma(x)), n).$$

Let  $S_f$  be the set of all random strings  $s'$  for which  $d(f) \leq \mathcal{A}(n, s', f) \leq \Delta(d(f))$ ; that is, it includes all the random strings that yield correct answers. Throughout this section and the next, we say that  $\mathcal{A}$   $\Delta$ -approximates  $d(f)$ . See (1). This should not be confused with the previous definition of  $\gamma$ -approximates  $d(f)$ . Here, we use the capital letter  $\Delta$  to prevent any ambiguity.

Let

$$\Psi_{\mathcal{A}}(d) = \mathbf{E}_{(f,s) \sim_u \text{Lin}(\mathbb{F},d) \times S(f)} [\mathcal{A}(n, s, f)]$$

where  $\sim_u$  indicates that  $f$  is chosen uniformly at random from  $\text{Lin}(\mathbb{F}, d)$  and  $s$  uniformly at random from  $S(f)$ . Since  $d \leq \mathcal{A}(n, s, f) \leq \Delta(d) \leq n$  for  $s \in S(f)$  where  $f \in \text{Lin}(\mathbb{F}, d)$ , we have

$$d \leq \Psi_{\mathcal{A}}(d) \leq \Delta(d) \leq n. \tag{2}$$

We note here that  $\Psi_{\mathcal{A}}(d)$  is independent of  $\delta$ , as in  $\mathcal{A}$ , we set  $\delta = \delta'$  for a fixed  $\delta'$ . This is crucial for ensuring the correctness of the proof. Also,  $\Psi_{\mathcal{A}}(d)$  depends on  $n$ . This will be essential only for the next result in the next section.

We first prove that the values of  $\Psi_{\mathcal{A}}(d)$  for  $d \in [n]$  can be approximated with high probability.

► **Lemma 7.** *Let  $0 < h < 1$ . Let  $\mathcal{A}$  be an algorithm that runs in time  $T(n)$ , uses  $Q(n)$  labeled examples of  $f \in \text{Lin}(\mathbb{F})$  according to the uniform distribution in the presence of random classification noise, and, with probability at least  $1 - \delta'$ ,  $\Delta$ -approximates  $d(f)$ . A table of real values  $\Psi'_{\mathcal{A}}(d)$  for  $1 \leq d \leq n$  can be constructed in time  $\tilde{O}(n^3/h^2)T(n) \log(1/\delta)$ , and without using any labeled examples. This table, with probability at least  $1 - \delta$ , satisfies  $|\Psi'_{\mathcal{A}}(d) - \Psi_{\mathcal{A}}(d)| \leq h$  for all  $d \in [n]$ .*

**Proof.** Define a random variable as the output  $D$  of the algorithm  $\mathcal{A}$ , obtained from running it on a uniformly random  $f$  from  $\text{Lin}(\mathbb{F}, d)$ , provided that the output lies within the interval  $[d, \Delta(d)]$ . The labeled examples of  $f$  can be generated by choosing a random uniform  $u \in \{0, 1\}^n$  and returning  $(u, f(u) + e)$  to  $\mathcal{A}$  where, with probability  $1 - \eta_b$ ,  $e = 0$  and, with probability  $\eta_b$ ,  $e$  is random uniform in  $\mathbb{F} \setminus \{0\}$ . Obviously,  $\mathbf{E}[D] = \Psi_{\mathcal{A}}(d)$ .

By Hoeffding's bound, to compute  $\mathbf{E}[D]$  with an additive error  $h$  and a confidence probability of at least  $1 - \delta/(2n)$ , we need to obtain  $t = O((n^2/h^2) \log(n/\delta))$  values of  $D$ . Since the success probability of obtaining a value of  $D$  in the interval  $[d, \Delta(d)]$  is  $1 - \delta' > 2/3$ , we need to run the algorithm  $O(t + \log(2n/\delta))$  times to acquire  $t$  values with a success probability at least  $1 - \delta/(2n)$ . Therefore, the time complexity is  $O((t + \log(2n/\delta))nT(n)) = O(tnT(n)) = \tilde{O}(n^3/h^2)T(n) \log(1/\delta)$ . ◀

Our next result shows how to estimate  $\Psi_{\mathcal{A}}(d(f))$  of the target  $f$  without knowing  $d(f)$ .

► **Lemma 8.** *Let  $0 < h < 1$  and  $\tau = O((n^2/h^2) \log(1/\delta))$ . Let  $\mathcal{A}$  be an algorithm that runs in time  $T(n)$ , uses  $Q(n)$  labeled examples of  $f \in \text{Lin}(\mathbb{F})$  according to the uniform distribution, in the presence of random classification noise, and, with probability at least  $1 - \delta'$ ,  $\Delta$ -approximates  $d(f)$ . There is an algorithm  $\mathcal{B}(n, h)$  that runs in time  $T' = \tau T(n)$ , uses  $Q' = \tau Q(n)$  labeled examples of  $f \in \text{Lin}(\mathbb{F})$  according to the uniform distribution in the presence of random classification noise and, with probability at least  $1 - \delta/2 - \tau\delta'$ , returns  $\psi$  that satisfies  $|\psi - \Psi_{\mathcal{A}}(d(f))| \leq h$ .*

**Proof.** Suppose  $v \in (\mathbb{F} \setminus \{0\})^n$  is chosen uniformly at random and  $\phi : [n] \rightarrow [n]$  is a uniformly random permutation. If we run  $\mathcal{A}$  with the target  $f = \lambda_1 x_{i_1} + \dots + \lambda_d x_{i_d}$ , and for every labeled example  $(a, b) \in \mathbb{F}^n \times \mathbb{F}$ , we modify the labeled example to  $((v_1^{-1} a_{\phi^{-1}(1)}, \dots, v_n^{-1} a_{\phi^{-1}(n)}), b)$ , then the new labeled examples remain uniform and consistent with the function  $g(x) = \lambda_1 v_{\phi(i_1)} x_{\phi(i_1)} + \dots + \lambda_d v_{\phi(i_d)} x_{\phi(i_d)}$ . This function  $g$  is a uniformly random element of  $\text{Lin}(\mathbb{F}, d)$ . Using this fact, we show how to approximate  $\Psi_{\mathcal{A}}(d)$ .

To this end, let  $\tau = O((n^2/h^2) \log(1/\delta))$ . We iterate  $\tau$  times, and at each iteration, we choose a random uniform  $v \in (\mathbb{F} \setminus \{0\})^n$  and random uniform permutation  $\phi : [n] \rightarrow [n]$ . We request for  $Q(n)$  labeled examples and modify each labeled example  $(a, b) \in \mathbb{F}^n \times \mathbb{F}$  to  $((v_1^{-1} a_{\phi^{-1}(1)}, \dots, v_n^{-1} a_{\phi^{-1}(n)}), b)$ . We then run  $\mathcal{A}$  on these labeled examples. Let  $D_i$  be the output of the  $i$ -th iteration. We then output  $\psi' = (\sum_{i=1}^{\tau} D_i) / \tau$ .

We now prove that, with probability at least  $1 - \delta/2 - \tau\delta'$ , we have  $|\psi' - \Psi_{\mathcal{A}}(d(f))| \leq h$ . Since  $\mathcal{A}(n)$  runs  $\tau$  times, with probability at least  $1 - \tau\delta'$ , all the seeds used by  $\mathcal{A}$  are in  $S(f)$  and  $d(f) \leq D_i \leq \Delta(d(f))$ . Also, since  $\mathcal{A}(n)$  runs on a uniformly random function in  $\text{Lin}(\mathbb{F}, d)$ , we have  $\mathbf{E}[D_i] = \Psi_{\mathcal{A}}(d)$ . By Hoeffding's bound, along with the fact that  $D_i \leq \Delta(d(f)) \leq n$ , we can conclude that, with probability at least  $1 - \delta/2$ , we have  $|\psi' - \Psi_{\mathcal{A}}(d(f))| \leq h$ .  $\blacktriangleleft$

Notice that in Lemma 8,  $\tau$  also depends on  $h$ . As  $h$  eventually will be  $O(1/n)$  and  $\delta' = 1/(|\mathbb{F}|n^7)$ , the success probability  $1 - \delta/2 - \tau\delta'$  will be  $1 - o_n(1)$  for  $\delta = 1/n$ .

We now show that in any large enough sub-interval of  $[0, n]$ , there is  $k$  for which  $\mathcal{A}$  can be used to learn  $\text{Lin}(\mathbb{F}, k)$ .

► **Lemma 9.** *Let  $\mathcal{A}(n)$  be an algorithm that runs in time  $T(n)$ , uses  $Q(n)$  labeled examples of  $f \in \text{Lin}(\mathbb{F})$  according to the uniform distribution in the presence of random classification noise, and, with probability at least  $1 - \delta'$ ,  $\Delta$ -approximates  $d(f)$ . For every  $1 \leq m \leq \min\{j | \Delta(j) = n\} = \gamma^{-1}(\gamma^{-1}(n))$  there exists  $m \leq k \leq \Delta(m) + 1$  and*

1. *An algorithm that, for every  $f \in \text{Lin}(\mathbb{F}, k)$ , with probability at least  $1 - \delta/8 - 2\tau n\delta'$ , where  $\tau = O(n^4 \log(1/\delta))$ , identifies the relevant variables of  $f$  from random uniform labeled examples in the presence of random classification noise. This algorithm runs in time  $\tilde{O}(n^5)T(n) \log(1/\delta)$  and uses  $\tilde{O}(n^4)Q(n) \log(1/\delta)$  labeled examples.*
2. *An algorithm that, with probability at least  $1 - \delta/2 - |\mathbb{F}|kn\delta'$ , properly learns  $\text{Lin}(\mathbb{F}, k)$ , from random uniform labeled examples in the presence of random classification noise. This algorithm runs in time  $\tilde{O}(|\mathbb{F}|kn^5)T(n) \log(1/\delta)$  and uses  $\tilde{O}(n^4)Q(n) \log(|\mathbb{F}|/\delta)$  labeled examples.*

Such  $k$  can be found in time  $\tilde{O}(n^5)T(n) \log(1/\delta)$ .

**Proof.** We first prove the result when the field is not the binary field. Let  $m$  be any integer such that  $1 \leq m \leq \min\{j | \Delta(j) = n\}$ . Since by (2),

$$\begin{aligned} \sum_{i=m}^{\Delta(m)} \Psi_{\mathcal{A}}(i+1) - \Psi_{\mathcal{A}}(i) &= \Psi_{\mathcal{A}}(\Delta(m)+1) - \Psi_{\mathcal{A}}(m) \\ &\geq \Delta(m) + 1 - \Delta(m) = 1, \end{aligned}$$

there is  $k$  such that  $m \leq k \leq \Delta(m)$  and

$$\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k) \geq \frac{1}{\Delta(m) - m + 1} \geq \frac{1}{n}.$$

First, we find such  $k$ . By Lemma 7, taking  $h = 1/(16n)$ , with probability at least  $1 - \delta/4$ , we can find  $k$  such that  $\Psi_{\mathcal{A}}(k+1) - \Psi_{\mathcal{A}}(k) \geq 7/(8n)$  in time  $\tilde{O}(n^5)T(n) \log(1/\delta)$ .

We now present an algorithm that learns  $\text{Lin}(\mathbb{F}, k)$ . The algorithm uses the algorithm in Lemma 8 to approximate  $\Psi_{\mathcal{A}}(d(f + x_i))$  and  $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$  for some  $\alpha \in \mathbb{F} \setminus \{0, 1\}$  and all  $i \in [n]$  with an additive error of  $1/(8n)$ . If  $x_i$  is not a relevant variable of the target function  $f$ , then both  $f + x_i$  and  $f + \alpha x_i$  are in  $\text{Lin}(\mathbb{F}, k + 1)$ . Consequently, we obtain two values in the interval  $[\Psi_{\mathcal{A}}(k + 1) - 1/(8n), \Psi_{\mathcal{A}}(k + 1) + 1/(8n)]$ . If  $x_i$  is a relevant variable in the function, then one of the functions, either  $f + x_i$  or  $f + \alpha x_i$  is in  $\text{Lin}(\mathbb{F}, k)$ , and therefore, one of the values is in the interval  $[\Psi_{\mathcal{A}}(k) - 1/(8n), \Psi_{\mathcal{A}}(k) + 1/(8n)]$ . Since  $\Psi_{\mathcal{A}}(k) + 1/(8n) < \Psi_{\mathcal{A}}(k + 1) - 1/(8n)$ , the intervals are disjoint, and thus we can distinguish between the two cases.

By Lemma 8, with probability  $1 - \delta/2 - \tau\delta'$ , we can approximate each  $\Psi_{\mathcal{A}}(d(f + x_i))$  (or  $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$ ) with an additive error  $h = 1/(8n)$  in time  $\tau T(n)$  and  $\tau Q(n)$  labeled examples, where  $\tau = O(n^4 \log(1/\delta))$ . Taking  $\delta/(8n)$  for  $\delta$ , with probability  $1 - \delta/8 - 2\tau n\delta'$ , we can approximate all  $\Psi_{\mathcal{A}}(d(f + x_i))$  and  $\Psi_{\mathcal{A}}(d(f + \alpha x_i))$ ,  $i \in [n]$  with an additive error  $h = 1/(8n)$  in time  $\tau' n T(n)$  and  $\tau' Q(n)$  labeled examples where  $\tau' = O(n^4 \log(n/\delta))$ . This completes the proof of item 1 for the case where the field is not the binary field.

To prove item 2, suppose, without loss of generality, that  $x_1, \dots, x_k$  are the relevant variables in  $f$ . We approximate  $\psi_{\alpha, i} := \Psi_{\mathcal{A}}(d(f - \alpha x_i + x_{k+1}))$  for all  $\alpha \in \mathbb{F}$  and for every  $i \in [n]$ . The result follows from the fact that  $\psi_{\alpha, i} = \Psi_{\mathcal{A}}(k)$  if and only if the coefficient of  $x_i$  is  $\alpha$ . Otherwise,  $\psi_{\alpha, i} = \Psi_{\mathcal{A}}(k + 1)$ . By Lemma 8, to approximate all  $\psi_{\alpha, i}$ , with success probability of  $1 - \delta/4 - |\mathbb{F}|kn\delta'$ , we need time  $O(|\mathbb{F}|kn^5 T(n) \log(|\mathbb{F}|n/\delta))$  and  $O(n^4 T(n) \log(|\mathbb{F}|n/\delta))$  labeled examples. This completes the proof of item 2 for the case where the field is not the binary field.

Similar to the approach described above, for the binary field, we can show that there exists a  $k$  such that  $\Psi_{\mathcal{A}}(k + 1) - \Psi_{\mathcal{A}}(k - 1) \geq 1/n$ . Then, use the algorithm described in Lemma 8 to approximate  $\Psi_{\mathcal{A}}(d(f + x_i))$  for all  $i \in [n]$ . If  $x_i$  is not a relevant variable of the target function  $f$ , then  $\Psi_{\mathcal{A}}(d(f + x_i)) \in [\Psi_{\mathcal{A}}(k + 1) - 1/(8n), \Psi_{\mathcal{A}}(k + 1) + 1/(8n)]$ . If  $x_i$  is a relevant variable in  $f$ , then  $\Psi_{\mathcal{A}}(d(f + x_i)) \in [\Psi_{\mathcal{A}}(k - 1) - 1/(8n), \Psi_{\mathcal{A}}(k - 1) + 1/(8n)]$ . Since both intervals are disjoint, we obtain the desired result.<sup>10</sup> ◀

Notice that in item 2, the success probability  $1 - \delta/2 - |\mathbb{F}|kn\delta'$ , and the time complexity depends on  $|\mathbb{F}|$ . We now present an alternative algorithm for finding the coefficients of the linear function, given that the algorithm knows the relevant variables.

► **Lemma 10.** *Let  $\mathcal{A}$  be an algorithm that, for every  $f \in \text{Lin}(\mathbb{F}, k)$ , runs in time  $T$ , uses  $Q$  random uniform labeled examples in the presence of random classification noise, and identifies the relevant variables of  $f$ . Then there is an algorithm that properly learns  $\text{Lin}(\mathbb{F}, k)$  in time  $T + \tilde{O}((n^3/(1 - \eta_b)^k + n/(1 - \eta_b - 1/|\mathbb{F}|)^2) \log(1/\delta))$  and uses  $Q + O(((1/(1 - \eta_b)^k + n/(1 - \eta_b - 1/|\mathbb{F}|)^2) \log(1/\delta)))$  random uniform labeled examples in the presence of random classification noise.*

**Proof.** We run  $\mathcal{A}$  to find the relevant variables. The algorithm that finds the coefficients iterates  $O((1/(1 - \eta_b)^k) \log(1/\delta))$  times. At each iteration, it requests  $k$  labeled examples and uses Gaussian elimination to find the coefficients. Then, it tests whether the output function matches the target. If not, it proceeds to the next iteration.

<sup>10</sup> Another approach is to utilize the fact that  $\Psi_{\mathcal{A}}(k) - \Psi_{\mathcal{A}}(k - 1) \geq 1/n$ , and for every  $i$ , approximate  $\Psi_{\mathcal{A}}(g_i)$ , where  $g_i = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$ , using all labeled examples  $(a, b)$  that satisfy  $a_i = 0$ .

### 38:12 Approximating the Number of Relevant Variables in a Parity Implies Proper Learning

Since  $\eta = \eta_b$ , the probability that all the labeled examples are correct is  $(1 - \eta_b)^k$ . If  $|\mathbb{F}| = q$ , the probability that  $k$  random entries in the examples form a non-singular matrix is

$$\left(1 - \frac{1}{q^k}\right) \left(1 - \frac{1}{q^{k-1}}\right) \cdots \left(1 - \frac{1}{q}\right) \geq \frac{1}{4}.$$

Therefore, after  $t = O((1/(1 - \eta_b)^k) \log(1/\delta))$  iterations, with probability at least  $1 - \delta/3$ , at least one of the outputs is the target. By Lemma 3, learning the target from a set of  $t$  linear functions with a success probability of  $1 - \delta/3$  requires  $\tilde{O}((1/(1 - \eta_b - 1/q)^2)(k + \log(1/\delta)))$  labeled examples.  $\blacktriangleleft$

We are now ready to prove Theorem 6.

**Proof.** Let  $\mathcal{A}'$  be an algorithm that runs in polynomial time, uses labeled examples according to the uniform distribution in the presence of random classification noise, and outputs  $\gamma^{-1}(d(f)) \leq D \leq \gamma(d(f))$ . Modify the algorithm to output  $\min(\gamma(D), n)$ . The algorithm now is a  $\Delta$ -approximation algorithm, where  $\Delta(x) = \min(\gamma(\gamma(x)), n)$ . Let  $m(n) = \gamma^{-1}(\gamma^{-1}(\pi(n)))$ . Since  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  is strictly increasing and is defined for all  $\mathbb{R}^+$ , we have  $\gamma^{-1} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ , is also strictly increasing, and  $m(n) = \omega_n(1)$ . Let  $\delta' = 1/(|\mathbb{F}|n^7)$  and  $\delta = 1/n$ . By Lemma 9, item 1, there exists  $m(n) \leq k(n) \leq \Delta(m(n)) = \pi(n)$ , and an algorithm that, for every  $f \in \text{Lin}(\mathbb{F}, k(n))$ , with probability at least  $1 - o_n(1) > 2/3$ , identifies the relevant variables of  $f$  from random uniform labeled examples in the presence of random classification noise. Also, this algorithm runs in polynomial time. Since  $k(n) \leq \pi(n)$ , by Lemma 10 and item 2 in Lemma 9, there is a  $\text{poly}(n, 1/(1 - \eta_b - 1/|\mathbb{F}|), \min(|\mathbb{F}|, 1/(1 - \eta_b)^{\pi(n)}))$  time learning algorithm for  $\text{Lin}(\mathbb{F}, k)$ . Since  $k(n) \geq m(n) = \omega_n(1)$ , the result follows.  $\blacktriangleleft$

## 3.2 Approximation Implies Learning $\text{Lin}(\mathbb{F})$

In this section, we prove.

**► Theorem 11.** *Let  $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be any strictly increasing function, where  $\gamma(x) > x$  for every  $x$ . Let  $\Gamma(x) := \gamma(\gamma(x))$ . Consider any  $T(n)$ -time algorithm  $\mathcal{A}'(n)$  that, for any linear function  $f \in \text{Lin}(\mathbb{F})$ , uses  $Q(n)$  random uniform labeled examples of  $f$  in the presence of random classification noise and, with probability at least  $2/3$ , returns a  $\gamma$ -approximation of the number of relevant variables  $d(f)$  of  $f$ . From  $\mathcal{A}'(n)$ , one can, in polynomial time, construct a  $\tilde{O}(|\mathbb{F}|\Gamma(n)^{12})T(O(\Gamma(n)^2)) \log(1/\delta)$ -time algorithm that properly learns  $\text{Lin}(\mathbb{F})$  from  $\tilde{O}(\Gamma(n)^8)Q(O(\Gamma(n)^2)) \log(|\mathbb{F}|/\delta)$  random uniform labeled examples in the presence of random classification noise.*

We first show that for every  $d$  satisfying  $12\Gamma(d)^2 \leq n$ , there exists a learning algorithm for  $\text{Lin}(\mathbb{F}, d)$ .

**► Lemma 12.** *Suppose that for every  $1 \leq m \leq m' := \Gamma^{-1}(n)$ , there exists a  $k$  such that  $m \leq k \leq \Gamma(m)$ , and an algorithm that runs in time  $T(n)$  and, with probability at least  $2/3$ , properly learns  $f \in \text{Lin}(\mathbb{F}, k)$  under the uniform distribution in the presence of random classification noise, and uses  $Q(n)$  labeled examples. Then, for every  $d$  such that  $12\Gamma(d)^2 \leq n$ , there is an algorithm that runs in time  $O(T(n) \log(1/\delta))$  and properly learns  $\text{Lin}(\mathbb{F}, d)$  under the uniform distribution in the presence of random classification noise, using  $O(Q(n) \log(1/\delta))$  labeled examples.*

**Proof.** Let  $d$  be an integer such that  $12\Gamma(d)^2 < n$ . Since  $\Gamma(d) < n$ , we have  $d \leq m'$ . Consequently, there exists  $k$  such that  $d \leq k \leq \Gamma(d) \leq \Gamma(m') = n$ , along with a proper learning algorithm  $\mathcal{B}(n, k)$  for  $\text{Lin}(\mathbb{F}, k)$ , that runs in time  $T(n)$  and uses  $Q(n)$  labeled examples.

We now present an algorithm for learning  $\text{Lin}(\mathbb{F}, d)$ . We uniformly at random draw  $k - d$  variables  $x_{i_1}, \dots, x_{i_{k-d}}$  and run the algorithm  $\mathcal{B}(n, k)$ . For each labeled example  $(a, b)$  of  $f$ , we feed  $\mathcal{B}$  with the labeled example  $(a, b + a_{i_1} + \dots + a_{i_{k-d}})$ . This modified labeled example serves as a labeled example for the function  $g = f + x_{i_1} + \dots + x_{i_{k-d}}$ . The probability that  $g \in \text{Lin}(\mathbb{F}, k)$  is the probability that none of the variables  $x_{i_1}, \dots, x_{i_{k-d}}$  are relevant in  $f$ . This probability is given by

$$\prod_{i=d}^k \left(1 - \frac{i}{n}\right) \geq 1 - \frac{k^2}{n} \geq 1 - \frac{\Gamma(d)^2}{n} \geq \frac{11}{12}.$$

Therefore, with probability at least  $1 - (1/3 + 1/12) > 1/2$ , algorithm  $\mathcal{B}(n, k)$  learns  $g$  and thus learns  $f$ . By Lemma 5, the result follows.  $\blacktriangleleft$

We now show how to construct a learning algorithm for  $\text{Lin}(\mathbb{F}, d)$  for every  $d \leq n$ .

**► Lemma 13.** *Suppose that for every  $n$  and every  $d$  that satisfies  $12\Gamma(d)^2 \leq n$ , there is an algorithm  $\mathcal{A}(n)$  that runs in time  $T(n)$  and, with probability at least  $2/3$ , properly learns  $f \in \text{Lin}(\mathbb{F}, d)$  under the uniform distribution in the presence of random classification noise, using  $Q(n)$  labeled examples. Let  $N(n) = 12\Gamma(n)^2$ . Then, for every  $n$  and every  $d \leq n$ , there is an algorithm that runs in time  $T(N(n))$  and, with probability at least  $2/3$ , properly learns  $f \in \text{Lin}(\mathbb{F}, d)$  under the uniform distribution in the presence of random classification noise, using  $Q(N(n))$  labeled examples.*

**Proof.** Let  $N = N(n)$ . Then for every  $d \leq n$ , we have  $12\Gamma(d)^2 \leq 12\Gamma(n)^2 = N(n)$ . We run  $\mathcal{A}(N)$ . Whenever the algorithm requests a labeled example, we draw a labeled example  $(a, b) \in \mathbb{F}^n \times \mathbb{F}$ , append  $N - n$  random uniform entries to  $a$ , creating  $a'$ . We then provide  $(a', b)$  to  $\mathcal{A}(N)$ . The algorithm is effective for any  $d$  that satisfies  $12\Gamma(d)^2 \leq N = 12\Gamma(n)^2$  and, thereby, covers all  $d \leq n$ .  $\blacktriangleleft$

Theorem 11 now follows from Lemma 9, 12 and 13.

---

## References

- 1 Dana Angluin and Philip D. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1987. doi:10.1007/BF00116829.
- 2 Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. *IEEE Trans. Inf. Theory*, 60(10):6636–6645, 2014. doi:10.1109/TIT.2014.2340869.
- 3 Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021. doi:10.1145/3444942.
- 4 Arnab Bhattacharyya, Ameet Gadekar, Suprovat Ghoshal, and Rishi Saket. On the hardness of learning sparse parities. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22–24, 2016, Aarhus, Denmark*, volume 57 of *LIPICs*, pages 11:1–11:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.ESA.2016.11.
- 5 Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. On learning  $k$ -parities with and without noise. *CoRR*, abs/1502.05375, 2015. arXiv:1502.05375.

- 6 Arnab Bhattacharyya, Ameet Gadekar, and Ninad Rajgopal. Improved learning of  $k$ -parities. *Theor. Comput. Sci.*, 840:249–256, 2020. doi:10.1016/J.TCS.2020.08.025.
- 7 Arnab Bhattacharyya, Piotr Indyk, David P. Woodruff, and Ning Xie. The complexity of linear dependence problems in vector spaces. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 496–508. Tsinghua University Press, 2011. URL: <http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/33.html>.
- 8 Avrim Blum. On-line algorithms in machine learning. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, pages 306–325. Springer, 1996. doi:10.1007/BFB0029575.
- 9 Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 253–262, 1994. doi:10.1145/195058.195147.
- 10 Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993. doi:10.1007/3-540-48329-2\_24.
- 11 Avrim Blum, Lisa Hellerstein, and Nick Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *J. Comput. Syst. Sci.*, 50(1):32–40, 1995. doi:10.1006/JCSS.1995.1004.
- 12 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. doi:10.1145/792538.792543.
- 13 Nader H. Bshouty. Exact learning from an honest teacher that answers membership queries. *Theor. Comput. Sci.*, 733:4–43, 2018. doi:10.1016/J.TCS.2018.04.034.
- 14 Nader H. Bshouty and Lisa Hellerstein. Attribute-efficient learning in query and mistake-bound models. *J. Comput. Syst. Sci.*, 56(3):310–319, 1998. doi:10.1006/JCSS.1998.1571.
- 15 Harry Buhrman, David García-Soriano, and Arie Matsliah. Learning parities in the mistake-bound model. *Inf. Process. Lett.*, 111(1):16–21, 2010. doi:10.1016/J.IPL.2010.10.009.
- 16 Qi Cheng and Daqing Wan. Complexity of decoding positive-rate primitive reed-solomon codes. *IEEE Trans. Inf. Theory*, 56(10):5217–5222, 2010. doi:10.1109/TIT.2010.2060234.
- 17 Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem. *IEEE Trans. Inf. Theory*, 58(11):6935–6941, 2012. doi:10.1109/TIT.2012.2209198.
- 18 Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM J. Comput.*, 29(2):545–570, 1999. doi:10.1137/S0097539797323571.
- 19 Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Trans. Inf. Theory*, 49(1):22–37, 2003. doi:10.1109/TIT.2002.806118.
- 20 Vitaly Feldman. Attribute-efficient and non-adaptive learning of parities and DNF expressions. *J. Mach. Learn. Res.*, 8:1431–1460, 2007. doi:10.5555/1314498.1314547.
- 21 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput.*, 39(2):606–645, 2009. doi:10.1137/070684914.
- 22 Elena Grigorescu, Lev Reyzin, and Santosh S. Vempala. On noise-tolerant learning of sparse parities and related problems. In Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, and Thomas Zeugmann, editors, *Algorithmic Learning Theory - 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011. Proceedings*, volume 6925 of *Lecture Notes in Computer Science*, pages 413–424. Springer, 2011. doi:10.1007/978-3-642-24412-4\_32.



- 23 David Guijarro, Víctor Lavín, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Inf. Process. Lett.*, 70(5):233–239, 1999. doi:10.1016/S0020-0190(99)00063-0.
- 24 David Guijarro, Jun Tarui, and Tatsuie Tsukiji. Finding relevant variables in PAC model with membership queries. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory, 10th International Conference, ALT '99, Tokyo, Japan, December 6-8, 1999, Proceedings*, volume 1720 of *Lecture Notes in Computer Science*, page 313. Springer, 1999. doi:10.1007/3-540-46769-6\_26.
- 25 Thomas Hofmeister. An application of codes to attribute-efficient learning. In Paul Fischer and Hans Ulrich Simon, editors, *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Computer Science*, pages 101–110. Springer, 1999. doi:10.1007/3-540-49097-3\_9.
- 26 Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 629–638. ACM, 2008. doi:10.1145/1374376.1374466.
- 27 Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. doi:10.1145/293347.293351.
- 28 Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory, 17th Annual Conference on Learning Theory, COLT 2004, Banff, Canada, July 1-4, 2004, Proceedings*, volume 3120 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2004. doi:10.1007/978-3-540-27819-1\_16.
- 29 Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 378–389, 2005. doi:10.1007/11538462\_32.
- 30 Daniele Micciancio. Locally dense codes. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 90–97. IEEE Computer Society, 2014. doi:10.1109/CCC.2014.17.
- 31 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *CoRR*, abs/2401.03703, 2024. doi:10.48550/arXiv.2401.03703.
- 32 Ryuhei Uehara, Kensei Tsuchida, and Ingo Wegener. Optimal attribute-efficient learning of disjunction, parity and threshold functions. In Shai Ben-David, editor, *Computational Learning Theory, Third European Conference, EuroCOLT '97, Jerusalem, Israel, March 17-19, 1997, Proceedings*, volume 1208 of *Lecture Notes in Computer Science*, pages 171–184. Springer, 1997. doi:10.1007/3-540-62685-9\_15.
- 33 Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015. doi:10.1145/2728167.
- 34 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.
- 35 Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inf. Theory*, 43(6):1757–1766, 1997. doi:10.1109/18.641542.
- 36 Di Yan, Yu Yu, Hanlin Liu, Shuoyao Zhao, and Jiang Zhang. An improved algorithm for learning sparse parities in the presence of noise. *Theor. Comput. Sci.*, 873:76–86, 2021. doi:10.1016/J.TCS.2021.04.026.



# The Number of Random 2-SAT Solutions Is Asymptotically Log-Normal

**Arnab Chatterjee** ✉ 🏠

Department of Computer Science, TU Dortmund, Germany

**Amin Coja-Oghlan** ✉ 🏠

Department of Computer Science, TU Dortmund, Germany

**Noela Müller** ✉ 🏠

Department of Mathematics and Computer Science, Eindhoven University of Technology, Netherlands

**Connor Riddlesden** ✉ 🏠

Department of Mathematics and Computer Science, Eindhoven University of Technology, Netherlands

**Maurice Rolvien** ✉ 🏠

Department of Computer Science, TU Dortmund, Germany

**Pavel Zakharov** ✉ 🏠

Department of Computer Science, TU Dortmund, Germany

**Haodong Zhu** ✉ 🏠

Department of Mathematics and Computer Science, Eindhoven University of Technology, Netherlands

---

## Abstract

We prove that throughout the satisfiable phase, the logarithm of the number of satisfying assignments of a random 2-SAT formula satisfies a central limit theorem. This implies that the log of the number of satisfying assignments exhibits fluctuations of order  $\sqrt{n}$ , with  $n$  the number of variables. The formula for the variance can be evaluated effectively. By contrast, for numerous other random constraint satisfaction problems the typical fluctuations of the logarithm of the number of solutions are *bounded* throughout all or most of the satisfiable regime.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** satisfiability problem, 2-SAT, random satisfiability, central limit theorem

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.39

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2405.03302>

**Funding** *Amin Coja-Oghlan:* DFG CO 646/3, DFG CO 646/5 and DFG CO 646/6.

*Noela Müller:* NWO Gravitation grant NETWORKS-024.002.003.

*Pavel Zakharov:* DFG CO 646/6

*Haodong Zhu:* European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 945045 and NWO Gravitation project NETWORKS-024.002.003



© Arnab Chatterjee, Amin Coja-Oghlan, Noela Müller, Connor Riddlesden, Maurice Rolvien, Pavel Zakharov, and Haodong Zhu; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 39; pp. 39:1–39:15



Leibniz International Proceedings in Informatics

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

## 1 Introduction

### 1.1 Background and motivation

The quest for satisfiability thresholds has been a guiding theme of research into random constraint satisfaction problems [7, 17, 24]. But once the satisfiability threshold has been pinpointed a question of no less consequence is to determine the distribution of the number of satisfying assignments within the satisfiable phase [33]. Indeed, the number of solutions is intimately tied to phase transitions that affect the geometry of the solution space, which in turn impacts the computational nature of finding or sampling solutions [4, 15, 28]. However, few tools are currently available to count solutions of random problems. Where precise rigorous results exist (such as in random NAESAT or XORSAT), the proofs typically rely on the method of moments (e.g., [6, 26, 40, 41]). Yet a necessary condition for the success of this approach is that the problem in question exhibits certain symmetries, which are absent in many interesting cases [7, 20].

The aim of the present paper is to shed a closer light on the number of satisfying assignments in random 2-SAT, the simplest random CSP that lacks said symmetry properties. While the random 2-SAT satisfiability threshold has been known since the 1990s [19, 31], a first-order approximation to the number of satisfying assignments has been obtained only recently [5]. This timeline reflects the computational complexity of the respective questions. As is well known, deciding the satisfiability of a 2-CNF reduces to directed reachability, solvable in polynomial time [10].

By contrast, calculating the number of satisfying assignments  $Z(\Phi)$  of a 2-CNF  $\Phi$  is a #P-hard task [45]. Nonetheless, Monasson and Zecchina [36] put forward a delicate physics-inspired conjecture as to the exponential order of the number of satisfying assignments of random 2-CNFs. Achlioptas et al. [5] recently proved this conjecture. Their theorem provides a first-order, law-of-large-numbers approximation of the logarithm of the number of satisfying assignments. The present paper contributes a much more precise result, namely a central limit theorem. We show that throughout the satisfiable phase the logarithm of the number of satisfying assignments, suitably shifted and scaled, converges to a Gaussian. This is the first central limit theorem of this type for any random CSP.

Let  $\Phi = \Phi_{n,m}$  be a random 2-CNF on  $n$  Boolean variables  $x_1, \dots, x_n$  with  $m$  clauses, drawn independently and uniformly from all  $4\binom{n}{2}$  possible 2-clauses. Suppose that  $m \sim dn/2$  for a fixed real  $d > 0$ . Thus,  $d$  gauges the average number of clauses in which a variable  $x_i$  appears. The value  $d = 2$  marks the satisfiability threshold; hence,  $\Phi$  is satisfiable with high probability (“w.h.p.”) if  $d < 2$ , and unsatisfiable w.h.p. if  $d > 2$  [19, 31]. Achlioptas et al. [5] determined a function  $\phi(d) > 0$  such that for all  $d < 2$ , i.e., throughout the entire satisfiable phase we have

$$Z(\Phi) = \exp(n\phi(d) + o(n)) \quad \text{w.h.p. ,} \quad (1)$$

thereby determining the leading exponential order of  $Z(\Phi)$ .

However, (1) fails to identify the limiting distribution of  $Z(\Phi)$ . To be precise, since (1) shows that  $Z(\Phi)$  scales exponentially, we expect this random variable to exhibit *multiplicative* fluctuations. Therefore, the appropriate goal is to find the limiting distribution of the logarithm of this random variable, i.e., of  $\log Z(\Phi)$ . Indeed, physics intuition suggests that  $\log Z(\Phi)$  should be asymptotically Gaussian [34]. The main result of the present paper confirms this hunch. Specifically, letting  $\Gamma_{\eta(d)}$  be a Gaussian with mean 0 and standard deviation  $\eta(d) > 0$ , we prove that for all  $0 < d < 2$ ,  $\log Z(\Phi)$  satisfies

$$\mathbb{P} [\log Z(\Phi) - \mathbb{E}[\log Z(\Phi) \mid Z(\Phi) > 0] < z\sqrt{m}] \sim \mathbb{P} [\Gamma_{\eta(d)} < z] \quad (z \in \mathbb{R}). \quad (2)$$

The order  $\Theta(\sqrt{n})$  of fluctuations confirmed by (2) sets random 2-SAT apart from a large family of other random constraint satisfaction problems. For example, for random graph  $q$ -colouring with  $q \geq 3$  colours the log of the number of  $q$ -colourings *superconcentrates*, i.e., merely has *bounded* fluctuations throughout most of the regime where the random graph is  $q$ -colourable [12].<sup>1</sup> The same is true of random NAESAT, XORSAT and the symmetric perceptron [1, 11, 20, 40]. In each of these cases, certain fundamental symmetry properties (e.g., that the set of  $q$ -colourings remains invariant under permutations of the colours) enable the computation of the number of solutions via the method of moments. Random 2-SAT lacks the respective symmetry (as the set of satisfying assignments is not generally invariant under swapping “true” and “false”), and accordingly (2) establishes that the number of solutions fails to superconcentrate (for more details see [20]).

### 1.2 The main result

The formula for the standard deviation  $\eta(d)$  from (2) comes in terms of a fixed point equation on a space of probability measures. Thus, let  $\mathcal{P}(\mathbb{R}^2)$  be the set of all (Borel) probability measures on  $\mathbb{R}^2$ . For  $0 < d < 2$  and  $0 \leq t \leq 1$  we define an operator

$$\log\text{BP}_{d,t}^{\otimes} : \mathcal{P}(\mathbb{R}^2) \rightarrow \mathcal{P}(\mathbb{R}^2), \quad \rho \mapsto \hat{\rho} = \log\text{BP}_{d,t}^{\otimes}(\rho), \quad (3)$$

as follows. Let

$$(\xi_{\rho,i})_{i \geq 1}, (\xi'_{\rho,i})_{i \geq 1}, (\xi''_{\rho,i})_{i \geq 1}, \quad \xi_{\rho,i} = \begin{pmatrix} \xi_{\rho,i,1} \\ \xi_{\rho,i,2} \end{pmatrix}, \xi'_{\rho,i} = \begin{pmatrix} \xi'_{\rho,i,1} \\ \xi'_{\rho,i,2} \end{pmatrix}, \xi''_{\rho,i} = \begin{pmatrix} \xi''_{\rho,i,1} \\ \xi''_{\rho,i,2} \end{pmatrix}$$

be random vectors with distribution  $\rho$ , let  $\mathbf{d} \stackrel{\text{dist}}{=} \text{Po}(td)$ ,  $\mathbf{d}', \mathbf{d}'' \stackrel{\text{dist}}{=} \text{Po}((1-t)d)$  and let  $\mathbf{s}_i, \mathbf{s}'_i, \mathbf{s}''_i, \mathbf{r}_i, \mathbf{r}'_i, \mathbf{r}''_i$  for  $i \geq 1$  be uniformly random on  $\{\pm 1\}$ , all mutually independent. Then  $\hat{\rho}$  is the distribution of the vector

$$\begin{pmatrix} \sum_{i=1}^{\mathbf{d}} \mathbf{s}_i \log \left( \frac{1}{2} (1 + \mathbf{r}_i \tanh(\xi_{\rho,i,1}/2)) \right) + \sum_{i=1}^{\mathbf{d}'} \mathbf{s}'_i \log \left( \frac{1}{2} (1 + \mathbf{r}'_i \tanh(\xi'_{\rho,i,1}/2)) \right) \\ \sum_{i=1}^{\mathbf{d}} \mathbf{s}_i \log \left( \frac{1}{2} (1 + \mathbf{r}_i \tanh(\xi_{\rho,i,2}/2)) \right) + \sum_{i=1}^{\mathbf{d}''} \mathbf{s}''_i \log \left( \frac{1}{2} (1 + \mathbf{r}''_i \tanh(\xi''_{\rho,i,2}/2)) \right) \end{pmatrix}.$$

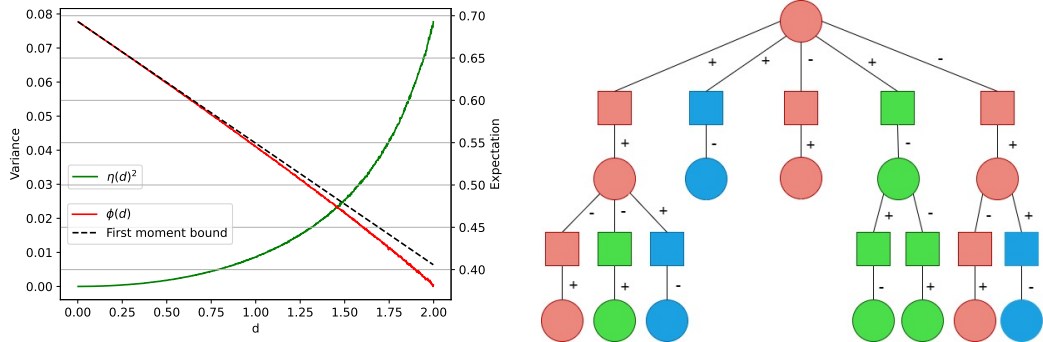
In addition, define a function  $\mathcal{B}_{d,t}^{\otimes} : \mathcal{P}(\mathbb{R}^2) \rightarrow (0, \infty]$  by letting

$$\mathcal{B}_{d,t}^{\otimes}(\rho) = \mathbb{E} \left[ \prod_{h=1}^2 \log \left( 1 - \frac{1}{4} (1 + \mathbf{r}_1 \tanh(\xi_{\rho,1,h}/2)) (1 + \mathbf{r}_2 \tanh(\xi_{\rho,2,h}/2)) \right) \right]. \quad (4)$$

► **Theorem 1.** *For any  $0 < d < 2$ ,  $t \in [0, 1]$ , there exists a unique probability measure  $\rho_{d,t} \in \mathcal{P}(\mathbb{R}^2)$  such that*

$$\rho_{d,t} = \log\text{BP}_{d,t}^{\otimes}(\rho_{d,t}) \quad \text{and} \quad \int_{\mathbb{R}^2} \|\xi\|_2^2 d\rho_{d,t}(\xi) < \infty. \quad (5)$$

<sup>1</sup> Formally, up to the so-called condensation threshold, which precedes the  $q$ -colourability threshold by a small additive constant, the logarithm of the number of  $q$ -colourings minus its expectation converges in distribution to a random variable with bounded moments [12, 13, 20].



**Figure 1** *Left:* Numerical approximations to the function  $\phi(d)$  from (1) (red) and the variance  $\eta(d)^2$  from (7) (green). The black dashed line is the first moment bound  $d \mapsto \log(2) + \frac{d}{2} \log(3/4)$ . *Right:* An illustration of the tree  $T^\otimes$  from Section 2.6.

Furthermore,

$$\lim_{n \rightarrow \infty} \frac{\log Z(\Phi) - \mathbb{E}[\log Z(\Phi) \mid Z(\Phi) > 0]}{\sqrt{m}} = \Gamma_{\eta(d)} \quad \text{in distribution, where} \quad (6)$$

$$\eta(d)^2 = \int_0^1 \mathcal{B}_{d,t}^\otimes(\rho_{d,t}) dt - \mathcal{B}_{d,0}^\otimes(\rho_{d,0}) \in (0, \infty). \quad (7)$$

The conditioning on  $\log Z(\Phi) > 0$  is necessary in (6), because even for  $d < 2$  the formula  $\Phi$  is unsatisfiable with probability  $\Omega(n^{-1})$ , in which case  $\log Z(\Phi) = -\infty$ . Moreover, the  $L^2$ -bound from (5) ensures that the integral (7) is well-defined. Finally, (6) implies (2).

How can the formula (7) be evaluated? Because the proof of the uniqueness of the stochastic fixed point  $\rho_{d,t}$  from (5) is based on the contraction method, a fixed point iteration will converge rapidly. In effect, for any  $d, t$  a discrete distribution that approximates  $\rho_{d,t}$  arbitrarily well (in Wasserstein distance) can be computed via a randomised algorithm called *population dynamics* [34, Chapter 14]. Since  $\mathcal{B}_{d,t}^\otimes(\rho_{d,t})$  varies continuously in  $d$  and  $t$ ,  $\eta(d)^2$  can thus be approximated within any desired accuracy, see Figure 1.

## 2 Proof strategy

The main challenge towards the proof of Theorem 1 is to get a handle on the variance of  $\log Z(\Phi)$  given satisfiability. The key idea, inspired by spin glass theory [18] but novel to random constraint satisfaction, is to count the *joint* number of satisfying assignments of two correlated random formulas. Once this is accomplished Theorem 1 will follow from the careful application of a general martingale central limit theorem. To get acclimatised we first revisit the method of moments, the reasons it fails on random 2-SAT and the combinatorial interpretation of the law of large numbers (1).

### 2.1 The method of moments fails

The default approach to estimating the number of solutions to a random CSP is the venerable second moment method [7]. Its thrust is to show that the second moment of the number of solutions is of the same order as the square of the expected number of solutions. If so then the moment computation together with small subgraph conditioning yields the precise limiting distribution of the number of solutions [23, 42]. However, this approach works only if the log of the number of solutions superconcentrates around the log of the expected number of solutions.

This necessary condition is not satisfied in random 2-SAT. In fact, a straightforward calculation yields

$$\frac{1}{n} \log \mathbb{E}[Z(\Phi)] \sim \log 2 + \frac{d}{2} \log(3/4). \quad (8)$$

The formula on the r.h.s. is displayed as the black dashed line in Figure 1. As can be verified analytically, this line strictly exceeds the function  $\phi(d)$  from (1) for any  $0 < d < 2$ . Consequently, (1) implies that  $\log Z(\Phi) \leq \log \mathbb{E}[Z(\Phi)] - \Omega(n)$  w.h.p. In other words, the expected number of solutions  $\mathbb{E}[Z(\Phi)]$  overshoots the typical number of solutions by an exponential factor w.h.p. ; cf. the discussion in [6, 8].

## 2.2 Belief Propagation

Instead of the method of moments, the prescription of the physics-based work of Monasson and Zecchina [36] is to estimate  $\log Z(\Phi)$  by way of the Belief Propagation (BP) message passing algorithm. This approach was vindicated rigorously by Achlioptas et al. [5].

As we will reuse certain elements of that analysis we dwell on BP briefly. For a clause  $a$  of a 2-CNF  $\Phi$  let  $\partial a = \partial_{\Phi} a$  be the set of variables that  $a$  contains. Moreover, for  $x \in \partial a$  let  $\text{sign}_{\Phi}(x, a) = \text{sign}(x, a) \in \{\pm 1\}$  be the sign with which  $x$  appears in  $a$ . Analogously, let  $\partial x = \partial_{\Phi} x$  be the set of clauses in which variable  $x$  appears. BP introduces “messages” between clauses  $a$  and the variables  $x \in \partial a$ . More precisely, each such clause-variable pair  $a, x$  comes with two messages  $\mu_{x \rightarrow a}, \mu_{a \rightarrow x}$ . The messages are probability distributions on “true” and “false”, which we represent by  $\pm 1$ . Thus,  $\mu_{x \rightarrow a}(\pm 1), \mu_{a \rightarrow x}(\pm 1) \geq 0$  and  $\mu_{x \rightarrow a}(1) + \mu_{x \rightarrow a}(-1) = \mu_{a \rightarrow x}(1) + \mu_{a \rightarrow x}(-1) = 1$ .

The messages get updated iteratively by an operator

$$\text{BP} : (\mu_{x \rightarrow a}, \mu_{a \rightarrow x})_{a, x \in \partial a} \mapsto (\hat{\mu}_{x \rightarrow a}, \hat{\mu}_{a \rightarrow x})_{a, x \in \partial a} = \text{BP}((\mu_{x \rightarrow a}, \mu_{a \rightarrow x})_{a, x \in \partial a}). \quad (9)$$

For a clause  $a$  with adjacent variables  $\partial a = \{x, y\}$  the updated messages  $\hat{\mu}_{a \rightarrow x}(\pm 1)$  are defined by

$$\hat{\mu}_{a \rightarrow x}(\text{sign}(x, a)) = \frac{1}{1 + \mu_{y \rightarrow a}(\text{sign}(y, a))}, \quad \hat{\mu}_{a \rightarrow x}(-\text{sign}(x, a)) = \frac{\mu_{y \rightarrow a}(\text{sign}(y, a))}{1 + \mu_{y \rightarrow a}(\text{sign}(y, a))}. \quad (10)$$

Moreover, for a variable  $x$  and a clause  $a \in \partial x$  we define<sup>2</sup>

$$\hat{\mu}_{x \rightarrow a}(s) = \frac{\prod_{b \in \partial x \setminus \{a\}} \mu_{b \rightarrow x}(s)}{\prod_{b \in \partial x \setminus \{a\}} \mu_{b \rightarrow x}(1) + \prod_{b \in \partial x \setminus \{a\}} \mu_{b \rightarrow x}(-1)} \quad (s \in \{\pm 1\}); \quad (11)$$

The purpose of BP is to heuristically “approximate” the marginal probabilities that a random satisfying assignment  $\sigma = \sigma_{\Phi}$  of  $\Phi$  will set a certain variable to a specific truth value. The “approximation” given by the set  $(\mu_{x \rightarrow a}, \mu_{a \rightarrow x})_{a, x \in \partial a}$  of messages reads

$$\mu_x(s) = \frac{\prod_{b \in \partial x} \mu_{b \rightarrow x}(s)}{\prod_{b \in \partial x} \mu_{b \rightarrow x}(1) + \prod_{b \in \partial x} \mu_{b \rightarrow x}(-1)} \quad (s \in \{\pm 1\}). \quad (12)$$

The BP “ansatz” now asks that we iterate the BP operator until an (approximate) fixed point is reached, i.e., ideally until  $\hat{\mu}_{a \rightarrow x} = \mu_{a \rightarrow x}$  and  $\hat{\mu}_{x \rightarrow a} = \mu_{x \rightarrow a}$  for all  $a, x$ . Then we evaluate the BP marginals (12) and plug them into a generic formula called the *Bethe free*

<sup>2</sup> For the sake of tidyness, if the above denominator vanishes we simply let  $\hat{\mu}_{x \rightarrow a}(\pm 1) = \frac{1}{2}$ .

entropy, which yields the BP “approximation” of  $\log Z(\Phi)$ ; an excellent exposition can be found in [34]. The BP recipe provably yields the correct result if the bipartite graph induced by the clause-variable incidences of the 2-CNF  $\Phi$  is acyclic, but may be totally off otherwise.

Of course, for  $1 < d < 2$  the bipartite graph associated with the random formula  $\Phi$  contains cycles in abundance. Nonetheless, (1) confirms that the BP formula provides a valid approximation to within  $o(n)$ . The proof is based on two observations. First, that the local structure of the clause-variable incidence graph can be described by a Galton-Watson tree. Second, that the Galton-Watson tree enjoys a spatial mixing property called *Gibbs uniqueness*.

Since the proof of Theorem 1 also harnesses Gibbs uniqueness, let us elaborate. To mimic the local structure of  $\Phi$  consider a multitype Galton-Watson tree  $\mathbf{T}$  whose types are *variable nodes* and *clause nodes* of four sub-types  $(s, s')$  with  $s, s' \in \{\pm 1\}$ . The root  $o$  is a variable node. The offspring of any variable node is a  $\text{Po}(d/4)$  number of clause nodes of each of the four sub-types. Finally, the offspring of a clause node is a single variable node. The clause type  $(s, s')$  indicates that  $s$  is the sign with which the parent variable appears in the clause, while  $s'$  determines the sign of the child variable. Thus, the Galton-Watson tree  $\mathbf{T}$  can be viewed as a (possibly infinite) 2-CNF. For an integer  $\ell \geq 0$  let  $\mathbf{T}^{(2\ell)}$  be the finite tree/2-CNF obtained by deleting all variables and clauses at a distance larger than  $2\ell$  from the root.

The tree  $\mathbf{T}$  approximates  $\Phi$  locally in the sense that for any fixed  $\ell$  and any given variable  $x_i$  the distribution of the depth- $2\ell$  neighbourhood of  $x_i$  in  $\Phi$  converges to  $\mathbf{T}^{(2\ell)}$  as  $n \rightarrow \infty$  (in the sense of local weak convergence). Moreover, Gibbs uniqueness posits that under random satisfying assignments of the tree-CNF  $\mathbf{T}^{(2\ell)}$  the truth value  $\sigma_o$  of the root under a random satisfying assignment  $\sigma$  decouples from the values  $\sigma_{\mathbf{T},y}$  of variables  $y \in \partial^{2\ell}o$  at distance precisely  $2\ell$  from  $o$  for large  $\ell$ . Formally, with  $S(\mathbf{T}^{(2\ell)})$  the set of satisfying assignments of the 2-CNF  $\mathbf{T}^{(2\ell)}$ , the following is true.

► **Proposition 2** ([5, Proposition 2.2]). *We have*

$$\lim_{\ell \rightarrow \infty} \mathbb{E} \left[ \max_{\tau \in S(\mathbf{T}^{(2\ell)})} \left| \mathbb{P} \left[ \sigma_o = 1 \mid \mathbf{T}^{(2\ell)}, \sigma_{\partial^{2\ell}o} = \tau_{\partial^{2\ell}o} \right] - \mathbb{P} \left[ \sigma_o = 1 \mid \mathbf{T}^{(2\ell)} \right] \right| \right] = 0. \quad (13)$$

### 2.3 Approaching the variance

The proof of the formula (1) combines the Gibbs uniqueness property and the local convergence to the Galton-Watson tree with a coupling argument called the “Aizenman-Sims-Starr scheme” [5]. Unfortunately, this combination does not seem precise enough to get a handle on the limiting distribution of  $\log Z(\Phi)$  by a long shot. Actually, it is anything but clear how even the *order* of the standard deviation of  $\log Z(\Phi)$  could be derived along these lines. One specific problem is that the rate of convergence of (13) diminishes as  $d$  approaches the satisfiability threshold.

To tackle this challenge we devise a combinatorial interpretation of  $\log^2 Z(\Phi)$ . A key idea, which we borrow from spin glass theory [18], is to set up a family of correlated random formulas. Specifically, given integers  $M, M' \geq 0$  we construct a correlated pair  $(\Phi_1(M, M'), \Phi_2(M, M'))$  of formulas on the variable set  $V_n = \{x_1, \dots, x_n\}$  as follows. Let  $(\mathbf{a}_i)_{i \geq 1}$ ,  $(\mathbf{a}'_i)_{i \geq 1}$ ,  $(\mathbf{a}''_i)_{i \geq 1}$  be sequences of mutually independent uniformly random clauses on  $V_n$ . Then

$$\begin{aligned} \Phi_1(M, M') &= \mathbf{a}_1 \wedge \dots \wedge \mathbf{a}_M \wedge \mathbf{a}'_1 \wedge \dots \wedge \mathbf{a}'_{M'}, \\ \Phi_2(M, M') &= \mathbf{a}_1 \wedge \dots \wedge \mathbf{a}_M \wedge \mathbf{a}''_1 \wedge \dots \wedge \mathbf{a}''_{M'}. \end{aligned} \quad (14)$$



Thus, the two formulas share clauses  $\mathbf{a}_1, \dots, \mathbf{a}_M$ . Additionally, each contains another  $M'$  independent clauses. In particular,  $\Phi_1(m, 0)$ ,  $\Phi_2(m, 0)$  are identical, while  $\Phi_1(0, m)$ ,  $\Phi_2(0, m)$  are independent.

Interpolating between these extreme cases offers a promising avenue for computing the variance: given that  $\Phi_1(M, m - M)$  and  $\Phi_2(M, m - M)$  are satisfiable for all  $M$ , we can write a telescoping sum

$$\begin{aligned} & \log Z(\Phi_1(m, 0)) \cdot \log Z(\Phi_2(m, 0)) - \log Z(\Phi_1(0, m)) \cdot \log Z(\Phi_2(0, m)) \\ &= \sum_{M=1}^m \log Z(\Phi_1(M, m - M)) \cdot \log Z(\Phi_2(M, m - M)) \\ & \quad - \log Z(\Phi_1(M - 1, m - M + 1)) \cdot \log Z(\Phi_2(M - 1, m - M + 1)). \end{aligned} \quad (15)$$

If we *could* take the expectation on the l.h.s. of (15), we would precisely obtain the variance of  $\log Z(\Phi)$ . Moreover, each summand on the r.h.s. amounts to a “local” change of swapping a shared clause for a pair of independent clauses. Yet we cannot just take the expectation of (15), because some  $\Phi_h(M, m - M)$  may be unsatisfiable. To remedy this, we will replace  $\log Z(\Phi)$  by a tamer random variable with the same limiting distribution. Its construction is based on the Unit Clause Propagation algorithm.

## 2.4 Unit Clause Propagation

Employed by all modern SAT solvers as a sub-routine, Unit Clause Propagation is a linear time algorithm that tracks the implications of partial assignments. The algorithm receives as input a 2-CNF  $\Phi$  along with a set  $\mathcal{L}$  of literals. These literals are deemed to be “true”. The algorithm then pursues direct logical implications, thereby identifying additional “implied” literals that need to be true so that no clause gets violated. This procedure is outlined in Steps 1–2 of Algorithm 1; the outcome of Steps 1–2 is independent of the order in which literals/clauses are processed.

■ **Algorithm 1** Pessimistic Unit Clause Propagation (“PUC”).

---

**Data:** A 2-CNF  $\Phi$  along with a set  $\mathcal{L}$  of literals deemed true.

- 1 **while** there exists a clause  $a \equiv l \vee \neg l'$  with  $l' \in \mathcal{L}$  and  $l \notin \mathcal{L}$  **do**
- 2     add literal  $l$  to  $\mathcal{L}$ ;
- 3 For variables  $x \in V(\Phi)$  such that  $x \in \mathcal{L}$  or  $\neg x \in \mathcal{L}$  let

$$\sigma_x = \begin{cases} 1 & \text{if } x \in \mathcal{L} \text{ and } \neg x \notin \mathcal{L}, \\ -1 & \text{if } \neg x \in \mathcal{L} \text{ and } x \notin \mathcal{L}, \\ 0 & \text{otherwise.} \end{cases}$$

---

Let  $\mathcal{C}$  be the set of all clauses  $a$  such that  $\sigma_x = 0$  for all  $x \in \partial a$  and return  $\mathcal{L}, \mathcal{C}, \sigma$ ;

---

Clearly, trouble brews if PUC ends up placing both a literal  $l$  and its negation  $\neg l$  into the set  $\mathcal{L}$ . Our “pessimistic” Unit Clause variant makes no attempt at mitigating such contradictions. Instead, Step 3 just constructs a partial assignment where all conflicting literals are set to a dummy value zero. Additionally, PUC identifies the set  $\mathcal{C}$  of *conflict clauses* that contain conflicted variables only.

## 39:8 The Number of Random 2-SAT Solutions Is Asymptotically Log-Normal

Now consider a 2-CNF  $\Phi$  on a set of variables  $V(\Phi)$ . For each possible literal  $l \in \{x, \neg x : x \in V(\Phi)\}$  we run  $\text{PUC}(\Phi, \mathcal{L} = \{l\})$ . Let  $\mathcal{C}(\Phi, \{l\})$  be the set of conflict clauses returned by PUC. Obtain the *pruned formula*  $\hat{\Phi}$  from  $\Phi$  by removing all clauses in  $\mathcal{C}(\Phi) = \bigcup_l \mathcal{C}(\Phi, \{l\})$ . Then it is easy to verify the following.

► **Fact 3.** *For any 2-CNF  $\Phi$  the pruned 2-CNF  $\hat{\Phi}$  is satisfiable.*

Generally, the pruned formula  $\hat{\Phi}$  could have far fewer clauses than the original formula  $\Phi$ . Accordingly, even if  $\Phi$  is satisfiable the number  $Z(\hat{\Phi})$  of satisfying assignments of  $\hat{\Phi}$  could dramatically exceed  $Z(\Phi)$ . However, the following proposition shows that on a random formula, the impact of pruning is modest.

► **Proposition 4.** *With probability  $1 - o(n^{-1/2})$  we have  $|\log Z(\hat{\Phi}) - \log Z(\Phi)| \leq n^{1/3}$ .*

### 2.5 Variance redux

The error bound from Proposition 4 is tight enough so that towards the proof of Theorem 1 it suffices to establish a central limit theorem for  $\log Z(\hat{\Phi})$ , i.e., the log of the number of satisfying assignments of the pruned formula. Once again the pivotal task to this end is to compute the variance of  $\log Z(\hat{\Phi})$ . Revisiting the telescoping sum (15), we obtain the following expression. Recalling (14), we write  $\hat{\Phi}_h(M, M') = \Phi_h(\widehat{M}, \widehat{M}')$  for the formula obtained by pruning  $\Phi_h(M, M')$ .

► **Lemma 5.** *Let*

$$\Delta(M) = \mathbb{E} \left[ \log \left( \frac{Z(\hat{\Phi}_1(M, m - M))}{Z(\hat{\Phi}_1(M - 1, m - M))} \right) \cdot \log \left( \frac{Z(\hat{\Phi}_2(M, m - M))}{Z(\hat{\Phi}_2(M - 1, m - M))} \right) \right], \quad (16)$$

$$\Delta'(M) = \mathbb{E} \left[ \log \left( \frac{Z(\hat{\Phi}_1(M - 1, m - M + 1))}{Z(\hat{\Phi}_1(M - 1, m - M))} \right) \cdot \log \left( \frac{Z(\hat{\Phi}_2(M - 1, m - M + 1))}{Z(\hat{\Phi}_2(M - 1, m - M))} \right) \right]. \quad (17)$$

$$\text{Then } \text{Var} \left[ \log Z(\hat{\Phi}) \right] = \sum_{M=1}^m (\Delta(M) - \Delta'(M)).$$

Lemma 5 expresses the variance as a sum of local changes. For example,  $\Phi_1(M, m - M)$  is obtained from  $\Phi_1(M - 1, m - M)$  by adding a single random clause, namely  $\mathbf{a}_M$ . Thus,  $\Delta(M)$  equals the expected change upon addition of a single *shared* clause – modulo the effect of pruning, that is.

But fortunately, on random formulas only a few clauses get pruned w.h.p. In effect, we can express the impact of these random changes neatly in terms of random satisfying assignments of the “small” formulas  $\hat{\Phi}_h(M - 1, m - M)$  that appear in (16)–(17). Specifically, the quotients in (16)–(17) boil down to the probabilities that random satisfying assignments of the “small” formulas survive the extra clause that gets added to obtain the 2-CNFs in the respective numerators. Thus, with  $\sigma = (\sigma_y)_{y \in V_n}$  denoting a random satisfying assignment of  $\hat{\Phi}_h(M - 1, m - M)$ , we obtain the following.

► **Proposition 6.** *Let  $1 \leq M \leq m$ . W.h.p. we have*

$$\begin{aligned} \frac{Z(\hat{\Phi}_h(M, m-M))}{Z(\hat{\Phi}_h(M-1, m-M))} &= \\ 1 - \prod_{y \in \partial \mathbf{a}_M} \mathbb{P} \left[ \sigma_y \neq \text{sign}(y, \mathbf{a}_M) \mid \hat{\Phi}_h(M-1, m-M), \mathbf{a}_M \right] + o(1) \quad (h = 1, 2), \\ \frac{Z(\hat{\Phi}_1(M-1, m-M+1))}{Z(\hat{\Phi}_1(M-1, m-M))} &= \\ 1 - \prod_{y \in \partial \mathbf{a}'_{m-M+1}} \mathbb{P} \left[ \sigma_y \neq \text{sign}(y, \mathbf{a}'_{m-M+1}) \mid \hat{\Phi}_1(M-1, m-M), \mathbf{a}'_{m-M+1} \right] + o(1), \\ \frac{Z(\hat{\Phi}_2(M-1, m-M+1))}{Z(\hat{\Phi}_2(M-1, m-M))} &= \\ 1 - \prod_{y \in \partial \mathbf{a}''_{m-M+1}} \mathbb{P} \left[ \sigma_y \neq \text{sign}(y, \mathbf{a}''_{m-M+1}) \mid \hat{\Phi}_2(M-1, m-M), \mathbf{a}'_{m-M+1} \right] + o(1). \end{aligned}$$

## 2.6 Local convergence in probability

To evaluate the expressions from Proposition 6 we need to get a grip on the *joint* distribution of the truth values of  $y$  under random satisfying assignments of the two correlated formulas  $\hat{\Phi}_h(M-1, m-M)$ . To this end we will devise a Galton-Watson tree  $\mathbf{T}^\otimes$  that mimics the *joint* distribution of the local structure of  $(\hat{\Phi}_1(M-1, m-M), \hat{\Phi}_2(M-1, m-M))$ . Subsequently, we will establish Gibbs uniqueness for this Galton-Watson tree to compute the expressions from Proposition 6.

The Galton-Watson tree  $\mathbf{T}$  from Section 2.2 that describes the local topology of the “plain” random formula  $\Phi$  had one type of variable nodes and four types  $(\pm 1, \pm 1)$  of clause nodes. To approach the correlated pair  $(\hat{\Phi}_1(M, m-M-1), \hat{\Phi}_2(M, m-M-1))$  we need a Galton-Watson process with three types of variable nodes and a full dozen types of clause nodes. Specifically, there are *shared*, *1-distinct* and *2-distinct* variable nodes. The root  $o$  of  $\mathbf{T}^\otimes$  is a shared variable node. The clause node types are  $(s, s')$ -*shared*,  $(s, s')$  *1-distinct* and  $(s, s')$  *2-distinct* for  $s, s' \in \{\pm 1\}$ .

In addition to  $d \in (0, 2)$  the offspring distributions of  $\mathbf{T}^\otimes = \mathbf{T}_{d,t}^\otimes$  involve a second parameter  $t \in [0, 1]$ :

- A *shared variable* spawns  $\text{Po}(dt/4)$  shared clauses of type  $(s, s')$  as well as  $\text{Po}(d(1-t)/4)$  1-distinct clauses of type  $(s, s')$  and  $\text{Po}(d(1-t)/4)$  2-distinct clauses of type  $(s, s')$  for any  $s, s' \in \{\pm 1\}$ .
- An *h-distinct variable* begets  $\text{Po}(d/4)$   $h$ -distinct clauses of type  $(s, s')$  for any  $s, s' \in \{\pm 1\}$  ( $h = 1, 2$ ).
- A *shared clause* has precisely one shared variable as its offspring.
- An *h-distinct clause* spawns a single  $h$ -distinct variable ( $h = 1, 2$ ).

Figure 1 provides an illustration of the tree  $\mathbf{T}^\otimes$ . Shared variables/clauses are indicated in red, 1-distinct variables/clauses in green and 2-distinct ones in blue.

From  $\mathbf{T}^\otimes$  we extract a pair  $(\mathbf{T}_1, \mathbf{T}_2)$  of correlated random trees. Specifically,  $\mathbf{T}_h$  is obtained from  $\mathbf{T}^\otimes$  by deleting all  $(3-h)$ -distinct variables and clauses. Hence, the parameter  $t$  determines how “similar”  $\mathbf{T}_1, \mathbf{T}_2$  are. Specifically, if  $t = 1$  then no  $\{1, 2\}$ -distinct clauses exist and thus  $\mathbf{T}_1, \mathbf{T}_2$  are identical. By contrast, if  $t = 0$  then  $\mathbf{T}_1, \mathbf{T}_2$  are independent copies of the tree  $\mathbf{T}$  from Section 2.2.

## 39:10 The Number of Random 2-SAT Solutions Is Asymptotically Log-Normal

For an integer  $\ell \geq 0$  obtain  $\mathbf{T}^{\otimes, (2\ell)}$ ,  $\mathbf{T}_1^{(2\ell)}$ ,  $\mathbf{T}_2^{(2\ell)}$  from  $\mathbf{T}^{\otimes}, \mathbf{T}_1, \mathbf{T}_2$  by omitting all nodes at a distance greater than  $2\ell$  from the root  $o$ . As in Section 2.2, we can interpret these trees as 2-CNFs, with the type  $(s, s')$  of a clause indicating the signs of its parent and child variables. We say that two possible outcomes  $T, T'$  of  $\mathbf{T}^{\otimes, (2\ell)}$  are *isomorphic* if there is a tree isomorphism that preserves the root  $o$  as well as all types.

Further, a variable  $x \in V_n$  is called a  $2\ell$ -instance of  $T$  in  $(\hat{\Phi}_1(M, M'), \hat{\Phi}_2(M, M'))$  if there exist isomorphisms  $\iota_h$  of the 2-CNFs  $T_h$  obtained from  $T$  by deleting all  $(3-h)$ -distinct variables/clauses to the depth- $2\ell$  neighbourhoods  $\partial_{\hat{\Phi}_h(M, M')}^{\leq 2\ell} x$  of  $x$  in  $\hat{\Phi}_h(M, M')$  such that

- the root gets mapped to  $x$ , i.e.,  $\iota_1(o) = \iota_2(o) = x$ ,
- for any shared variable  $y$  of  $T_1, T_2$  the image variables coincide, i.e.,  $\iota_1(y) = \iota_2(y)$ ,
- for any shared clauses  $a$  of  $T_1, T_2$  the image  $\iota_1(a) = \iota_2(a) \in \{\mathbf{a}_1, \dots, \mathbf{a}_M\}$  is a shared clause,
- for any 1-distinct clause  $a$  whose parent in  $T_1$  is a shared variable,  $\iota_1(a) \in \{\mathbf{a}'_1, \dots, \mathbf{a}'_{M'}\}$ , and
- for any 2-distinct clause  $a$  whose parent in  $T_2$  is a shared variable,  $\iota_1(a) \in \{\mathbf{a}''_1, \dots, \mathbf{a}''_{M'}\}$ .

Let  $\mathbf{N}^{(2\ell)}(T, (\Phi_1(M, M'), \Phi_2(M, M')))$  be the number of  $2\ell$ -instances of  $T$  in  $(\Phi_1(M, M'), \Phi_2(M, M'))$ . The following proposition confirms that  $\mathbf{T}^{\otimes}$  models the local structure of  $(\hat{\Phi}_1(M, M'), \hat{\Phi}_2(M, M'))$  faithfully.

► **Proposition 7.** *Let  $\ell > 0$  be a fixed integer, let  $t \in [0, 1]$  and suppose that  $M \sim tdn/2$  and  $M' \sim (1-t)dn/2$ . Then w.h.p. for all possible outcomes  $T$  of  $\mathbf{T}^{\otimes, (2\ell)}$  we have  $\mathbf{N}^{(2\ell)}(T, (\hat{\Phi}_1(M, M'), \hat{\Phi}_2(M, M'))) \sim n\mathbb{P}[\mathbf{T}^{\otimes, (2\ell)} \cong T]$ .*

### 2.7 Correlated Belief Propagation

Now that we have a branching process description of our pair of correlated formulas the next step is to run BP on the random trees  $(\mathbf{T}_1, \mathbf{T}_2)$  to find the joint distribution of the truth values  $\sigma_{\mathbf{T}_1^{(2\ell)}, o}, \sigma_{\mathbf{T}_2^{(2\ell)}, o}$  assigned to the root. Hence, let

$$\boldsymbol{\mu}^{(2\ell)} = \left( \mathbb{P} \left[ \sigma_{\mathbf{T}_1^{(2\ell)}, o} = 1 \mid \mathbf{T}^{\otimes} \right], \mathbb{P} \left[ \sigma_{\mathbf{T}_2^{(2\ell)}, o} = 1 \mid \mathbf{T}^{\otimes} \right] \right) \in (0, 1)^2. \quad (18)$$

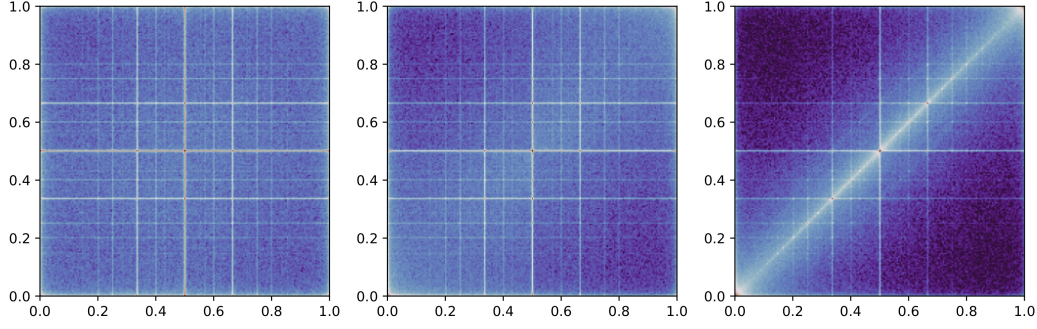
Since BP is exact on trees, we could calculate these marginals by iterating (9)–(11) for  $2\ell$  steps, starting from all-uniform messages. But our objective is not merely to calculate the marginals of a specific pair of trees, but the *distribution* of the vector (18) for a random  $\mathbf{T}^{\otimes}$ . Fortunately, due to the Markovian nature of the Galton-Watson tree  $\mathbf{T}^{\otimes}$ , the bottom-up BP computation on a random tree can be expressed by a fixed point iteration on the space of probability distributions on  $\mathbb{R}^2$ . The appropriate operator is the  $\log\text{BP}_{d,t}^{\otimes}$ -operator from (3). To be precise, that operator expresses the updates of the log-likelihood ratios of the BP messages from (10)–(11). Thus, let

$$\mathfrak{t} : (z_1, z_2) \in \mathbb{R}^2 \mapsto ((1 + \tanh(z_1/2))/2, (1 + \tanh(z_2/2))/2) \in (0, 1)^2$$

be the function that maps log-likelihood ratios back to probabilities. Furthermore, for a probability measure  $\rho \in \mathcal{P}(\mathbb{R}^2)$  let  $\mathfrak{t}(\rho)$  be the pushforward probability measure on  $(0, 1)^2$ .<sup>3</sup>

► **Proposition 8.** *Let  $\rho_{d,t}^{(0)} \in \mathcal{P}(\mathbb{R}^2)$  be the atom at the origin and let  $\rho_{d,t}^{(\ell)} = \log\text{BP}_{d,t}^{\otimes}(\rho_{d,t}^{(\ell-1)})$ . Then  $\boldsymbol{\mu}^{(2\ell)}$  has distribution  $\mathfrak{t}(\rho_{d,t}^{(\ell)})$ .*

<sup>3</sup> That is, for a measurable  $\mathfrak{A} \subseteq (0, 1)^2$  we have  $\mathfrak{t}(\rho)(\mathfrak{A}) = \rho(\mathfrak{t}^{-1}(\mathfrak{A}))$ .



■ **Figure 2** The distributions  $t(\rho_{d,t})$  for  $d = 1.9$  and  $t = 0.1, 0.5, 0.9$ .

We employ the contraction method to show that the sequence  $(\rho_{d,t}^{(\ell)})_{\ell \geq 1}$  of measures converges.

► **Proposition 9.** *There exists a unique  $\rho_{d,t} \in \mathcal{P}(\mathbb{R}^2)$  that satisfies (5) and  $\lim_{\ell \rightarrow \infty} \rho_{d,t}^{(\ell)} = \rho_{d,t}$  weakly.*

Furthermore, the Gibbs uniqueness property (13) extends to  $\mathbf{T}_1$  and  $\mathbf{T}_2$ .

► **Corollary 10.** *For all  $t \in [0, 1]$  and  $h = 1, 2$  we have*

$$\mathbb{E} \left[ \max_{\tau \in S(\mathbf{T}_h^{(2\ell)})} \left| \mathbb{P} \left[ \sigma_{\mathbf{T}_h^{(2\ell)}, o} = 1 \mid \mathbf{T}^{\otimes}, \sigma_{\mathbf{T}_h^{(2\ell)}, \partial^{2\ell} o} = \tau_{\partial^{2\ell} o} \right] - \mathbb{P} \left[ \sigma_{\mathbf{T}_h^{(2\ell)}, o} = 1 \mid \mathbf{T}^{\otimes} \right] \right| \right] \rightarrow 0, \tag{19}$$

as  $\ell \rightarrow +\infty$ .

Combining Propositions 8 and 9 and Corollary 10, we are now in a position to pinpoint the joint marginals of  $\hat{\Phi}_1(M, M')$ ,  $\hat{\Phi}_2(M, M')$ . Formally, let

$$\pi_{\hat{\Phi}_1(M, M'), \hat{\Phi}_2(M, M')} = \frac{1}{n} \sum_{i=1}^n \delta_{(\mathbb{P}[\sigma_{\hat{\Phi}_1(M, M'), x_i} = 1 \mid \hat{\Phi}_1(M, M')], \mathbb{P}[\sigma_{\hat{\Phi}_2(M, M'), x_i} = 1 \mid \hat{\Phi}_2(M, M')])} \in \mathcal{P}([0, 1]^2)$$

be the empirical distribution of the joint marginals of  $\hat{\Phi}_1(M, M')$  and  $\hat{\Phi}_2(M, M')$ , which we need to know to evaluate the expressions from Proposition 6. Furthermore, denote by  $W_1(\cdot, \cdot)$  the Wasserstein  $L^1$ -distance of two probability measures on  $[0, 1]^2$ .

► **Corollary 11.** *For any  $t \in [0, 1]$  and any  $M \sim \text{tnd}/2$ ,  $M' \sim (1 - t)dn/2$  we have*

$$\mathbb{E} \left[ W_1 \left( \pi_{\hat{\Phi}_1(M, M'), \hat{\Phi}_2(M, M')}, t(\rho_{d,t}) \right) \right] = o(1).$$

Finally, combining Proposition 6 with Corollary 11, we obtain the variance of  $\log Z(\hat{\Phi})$ .

► **Corollary 12.** *With  $\eta(d)^2$  from (7) we have  $\eta(d) > 0$  and  $\text{Var} \log Z(\hat{\Phi}) \sim m\eta_d^2$ .*

Because the proof of Proposition 9 is based on a contraction argument, for any  $d, t$  the distribution  $\rho_{d,t}$  can be approximated effectively within any given accuracy via a fixed point iteration. Figure 2 displays approximations to  $t(\rho_{d,t})$  for different values of  $t$  and shows how correlations between the two coordinates of the random vector increase with  $t$  (brighter diagonal).

## 2.8 The central limit theorem

With the variance computation done, we have now overcome the greatest hurdle en route to Theorem 1. Indeed, to obtain the desired asymptotic normality we just need to combine the techniques from the variance computation with a generic martingale central limit theorem.

To this end we set up a filtration  $(\mathfrak{F}_{n,M})_{0 \leq M \leq m_n}$  by letting  $\mathfrak{F}_{n,M}$  be the  $\sigma$ -algebra generated by  $\mathbf{a}_1, \dots, \mathbf{a}_M$ . Hence, conditioning on  $\mathfrak{F}_{n,M}$  amounts to conditioning on  $\mathbf{a}_1, \dots, \mathbf{a}_M$ , while averaging on the remaining clauses  $\mathbf{a}_{M+1}, \dots, \mathbf{a}_m$ . The conditional expectations

$$\mathbf{Z}_{n,M} = m^{-1/2} \mathbb{E} \left[ \log Z(\hat{\Phi}) \mid \mathfrak{F}_{n,M} \right] \quad (20)$$

then form a Doob martingale. Let  $\mathbf{X}_{n,M} = \mathbf{Z}_{n,M} - \mathbf{Z}_{n,M-1}$  be the martingale differences.

► **Proposition 13.** *For all  $0 < d < 2$  the martingale (20) satisfies*

$$\lim_{n \rightarrow \infty} \mathbb{E} \left[ \max_{1 \leq M \leq m} |\mathbf{X}_{n,M}| \right] = 0 \quad \text{and} \quad \lim_{n \rightarrow \infty} \mathbb{E} \left| \eta(d)^2 - \sum_{M=1}^m \mathbf{X}_{n,M}^2 \right| = 0. \quad (21)$$

Thanks to pruning, the first condition from (21) is easily checked. Furthermore, the steps that we pursued towards the proof of Corollary 12, i.e., the variance calculation, also imply the second condition without further ado. Finally, as (21) demonstrates that the marginal differences are small and that the variance process converges to a deterministic limit, Theorem 1 follows the general martingale central limit theorem from [27].

## 3 Discussion

The hunt for satisfiability thresholds of random constraint satisfaction problems was launched by the experimental work of Cheeseman, Kanefsky and Taylor [17]. The 2-SAT threshold was the first one to be caught [19, 31]. Subsequent successes include the 1-in- $k$ -SAT threshold [3] and the  $k$ -XORSAT threshold [26, 40]. Furthermore, Friedgut [29] proved the existence of non-uniform (i.e.,  $n$ -dependent) satisfiability thresholds in considerable generality. The plot thickened when physicists employed a compelling but non-rigorous technique called the cavity method to “predict” the exact satisfiability thresholds of many further problems, including the  $k$ -SAT problem for  $k \geq 3$  [35]. A line of rigorous work [6, 8, 22] culminated in the verification of this physics prediction for large  $k$  [24].

Even though the satisfiability threshold of random 2-SAT was determined already in the 1990s, the problem continued to receive considerable attention. For example, Bollobás, Borgs, Chayes, Kim and Wilson [14] investigated the scaling window around the satisfiability threshold, a point on which a recent contribution by Dovgal, de Panafieu and Ravelomanana elaborates [25]. Abbe and Montanari [2] made the first substantial step towards the study of the number of satisfying assignments that  $\frac{1}{n} \log Z(\Phi)$  converges in probability to a deterministic limit  $\varphi(d)$  for Lebesgue-almost all  $d \in (0, 2)$ . However, their techniques do not reveal the value  $\varphi(d)$ . Moreover, Montanari and Shah [37] obtain a “law-of-large-numbers” estimate of the number of assignments that violate all but  $o(n)$  clauses for  $d < 1.16$ . Finally, the aforementioned article of Achlioptas et al. [5] verifies the prediction from [36] as to the number of satisfying assignments for all  $d < 2$ . The main result of the present paper refines these results considerably by establishing a central limit theorem.

For random  $k$ -CNFs with  $k \geq 3$  an upper bound on the number of satisfying assignments can be obtained via the interpolation method from mathematical physics [39]. This bound matches the predictions of the cavity method [34]. However, no matching lower bound is

currently known. The precise physics prediction called the “replica symmetric solution” has only been verified for “soft” versions of random  $k$ -SAT where unsatisfied clauses are penalised but not strictly forbidden, and for clause-to-variable ratios well below the satisfiability threshold [37, 38, 44].

Random CSPs such as random  $k$ -XORSAT or random  $k$ -NAESAT that exhibit stronger symmetry properties than random  $k$ -SAT tend to be amenable to the method of moments [6].<sup>4</sup> Therefore, more is known about their number of solutions. For example, due to the inherent connection to linear algebra, the number of satisfying assignments of random  $k$ -XORSAT formulas is known to concentrate on a single value right up to the satisfiability threshold [11, 26, 40]. Furthermore, in random  $k$ -NAESAT, random graph colouring and several related problems, the logarithm of the number of solutions superconcentrates, i.e., has only bounded fluctuations for constraint densities up to the so-called condensation threshold, a phase transition that shortly precedes the satisfiability threshold [12, 20, 41]. The same is true of random  $k$ -SAT instances with regular literal degrees [23]. A further example is the symmetric perceptron [1], where the number of solutions superconcentrates but the limiting distribution is a log-normal with bounded variance. Going beyond the condensation transition, Sly, Sun and Zhang [43] proved that the number of satisfying assignments of random regular  $k$ -NAESAT formulas matches the “1-step replica symmetry breaking” prediction from physics.

Apart from the superconcentration results for symmetric problems from [12, 23, 20, 41], the limiting distribution of the logarithm of the number of solutions has not been known in any random constraint satisfaction problem. In particular, Theorem 1 is the first central limit theorem for this quantity in any random CSP. We expect that the technique developed in the present work, particularly the use of two correlated random instances in combination with spatial mixing, can be extended to other problems. The present use of correlated instances is inspired by the work of Chen, Dey and Panchenko [18] on the  $p$ -spin model from mathematical physics, a generalisation of the famous Sherrington-Kirkpatrick model. That said, on a technical level the present use of correlated instances is quite different from the approach from [18]. Specifically, while here we construct correlated 2-CNFs that share a specific fraction of their clauses and employ a martingale central limit theorem, Chen, Dey and Panchenko combine a continuous interpolation of two mixed  $p$ -spin Hamiltonians with Stein’s method.

A further line of work deals with central limit theorems for random optimisation problems. Cao [16] provided a general framework based on the “objective method” [9]. Unfortunately, the conditions of Cao’s theorem tend to be unwieldy for MAX CSP problems with hard constraints. Recent work of Kreačič [32] and Glasgow, Kwan, Sah, Sawhney [30] on the matching number therefore instead resorts to the use of stochastic differential equations. A promising question for future work might be whether the present method of considering correlated instances might extend to random optimisation problems.

---

## References

- 1 E. Abbe, S. Li, and A. Sly. Proof of the contiguity conjecture and lognormal limit for the symmetric perceptron. In *Proc. 62nd FOCS*, pages 327–338, 2022.
- 2 E. Abbe and A. Montanari. On the concentration of the number of solutions of random satisfiability formulas. *RSA*, 45:362–382, 2014.

---

<sup>4</sup> Formally, by “symmetry” we mean that the empirical distribution of the marginals of random solutions converges to an atom; cf. [21].

- 3 D. Achlioptas, A. Chtcherba, G. Istrate, and C. Moore. The phase transition in 1-in- $k$  sat and  $n$ -ae 3-sat. In *Proc. 12th SODA*, pages 721–722, 2001.
- 4 D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. In *Proc. 49th FOCS*, pages 793–802, 2008.
- 5 D. Achlioptas, A. Coja-Oghlan, M. Hahn-Klimroth, J. Lee, N. Müller, M. Penschuck, and G. Zhou. The number of satisfying assignments of random 2-sat formulas. *Random Structures and Algorithms*, 58:609–647, 2021.
- 6 D. Achlioptas and C. Moore. Random  $k$ -sat: two moments suffice to cross a sharp threshold. *SIAM Journal on Computing*, 36:740–762, 2006.
- 7 D. Achlioptas, A. Naor, and Y. Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435:759–764, 2005.
- 8 D. Achlioptas and Y. Peres. The threshold for random  $k$ -sat is  $2^k \ln 2 - O(k)$ . *Journal of the AMS*, 17:947–973, 2004.
- 9 D. Aldous and J. Steele. The objective method: probabilistic combinatorial optimization and local weak convergence. In H. Kesten, editor, *Probability on Discrete Structures*. Springer, 2004.
- 10 B. Aspövall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8:121–123, 1979.
- 11 P. Ayre, A. Coja-Oghlan, P. Gao, and N. Müller. The satisfiability threshold for random linear equations. *Combinatorica*, 40:179–235, 2020.
- 12 V. Bapst, A. Coja-Oghlan, and C. Efthymiou. Planting colourings silently. *Combinatorics, Probability and Computing*, 26:338–366, 2017.
- 13 V. Bapst, A. Coja-Oghlan, S. Hetterich, F. Rassmann, and D. Vilenchik. The condensation phase transition in random graph coloring. *Communications in Mathematical Physics*, 341:543–606, 2016.
- 14 B. Bollobás, C. Borgs, J. Chayes, J. Kim, and D. Wilson. The scaling window of the 2-sat transition. *RSA*, 18:201–256, 2001.
- 15 G. Bresler and B. Huang. The algorithmic phase transition of random  $k$ -sat for low degree polynomials. In *Proc. 62nd FOCS*, pages 298–309, 2021.
- 16 S. Cao. Central limit theorems for combinatorial optimization problems on sparse erdős-rényi graphs. *Annals of Applied Probability*, 31:1687–1723, 2021.
- 17 P. Cheeseman, B. Kanefsky, and W. Taylor. Where the *really* hard problems are. In *Proc. IJCAI*, pages 331–337, 1991.
- 18 W.-K. Chen, P. Dey, and D. Panchenko. Fluctuations of the free energy in the mixed p-spin models with external field. *Probability Theory and Related Fields*, 168:41–53, 2017.
- 19 V. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *Proc. 33th FOCS*, pages 620–627, 1992.
- 20 A. Coja-Oghlan, T. Kapetanopoulos, and N. Müller. The replica symmetric phase of random constraint satisfaction problems. *Combinatorics, Probability and Computing*, 29:346–422, 2020.
- 21 A. Coja-Oghlan, F. Krzakala, W. Perkins, and L. Zdeborová. Information-theoretic thresholds from the cavity method. *Advances in Mathematics*, 333:694–795, 2018.
- 22 A. Coja-Oghlan and K. Panagiotou. The asymptotic  $k$ -sat threshold. *Advances in Mathematics*, 288:985–1068, 2016.
- 23 A. Coja-Oghlan and N. Wormald. The number of satisfying assignments of random regular  $k$ -sat formulas. *Combinatorics, Probability and Computing*, 27:496–530, 2018.
- 24 J. Ding, A. Sly, and N. Sun. Proof of the satisfiability conjecture for large  $k$ . *Annals of Mathematics*, 196:1–388, 2022.
- 25 S. Dovgal, É. de Panafieu, and V. Ravelomanana. Exact enumeration of satisfiable 2-sat formulae. *arXiv:2108.08067*, 2021. [arXiv:2108.08067](https://arxiv.org/abs/2108.08067).
- 26 O. Dubois and J. Mandler. The 3-xorsat threshold. In *Proc. 43rd FOCS*, pages 769–778, 2002.
- 27 G. Eagleson. Martingale convergence to mixtures of infinitely divisible laws. *Annals of Probability*, 3:557–562, 1975.



- 28 C. Efthymiou. On sampling symmetric gibbs distributions on sparse random graphs and hypergraphs. In *Proc. 49th ICALP*, page 57, 2022.
- 29 E. Friedgut. Sharp thresholds of graph properties, and the  $k$ -sat problem. *Journal of the AMS*, 12:1017–1054, 1999.
- 30 M. Glasgow, M. Kwan, A. Sah, and M. Sawhney. A central limit theorem for the matching number of a sparse random graph. *arXiv:2402.05851*, 2024.
- 31 A. Goerdt. A threshold for unsatisfiability. *Journal of Computer and System Sciences*, 53:469–486, 1996.
- 32 E. Kreačić. *Some problems related to the Karp-Sipser algorithm on random graphs*. PhD thesis, University of Oxford, 2017.
- 33 F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104:10318–10323, 2007.
- 34 M. Mézard and A. Montanari. *Information, physics and computation*. Oxford University Press, 2009.
- 35 M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297:812–815, 2002.
- 36 R. Monasson and R. Zecchina. The entropy of the  $k$ -satisfiability problem. *Physical Review Letters*, 76:3881, 1996.
- 37 A. Montanari and D. Shah. Counting good truth assignments of random  $k$ -sat formulae. In *Proc. 18th SODA*, pages 1255–1264, 2007.
- 38 D. Panchenko. On the replica symmetric solution of the  $K$ -sat model. *Electronic Journal of Probability*, 19:67, 2014.
- 39 D. Panchenko and M. Talagrand. Bounds for diluted mean-fields spin glass models. *Probability Theory and Related Fields*, 130:319–336, 2004.
- 40 B. Pittel and G. Sorkin. The satisfiability threshold for  $k$ -xorsat. *Combinatorics, Probability and Computing*, 25:236–268, 2016.
- 41 F. Rassmann. On the number of solutions in random graph  $k$ -colouring. *Combinatorics, Probability and Computing*, 28:130–158, 2019.
- 42 R. Robinson and N. Wormald. Almost all regular graphs are hamiltonian. *Random Structures and Algorithms*, 5:363–374, 1994.
- 43 A. Sly, N. Sun, and Y. Zhang. The number of solutions for random regular nae-sat. *Probability Theory and Related Fields*, 182:1–109, 2022.
- 44 M. Talagrand. The high temperature case for the random  $K$ -sat problem. *Probability Theory and Related Fields*, 119:187–212, 2001.
- 45 L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.




# Private Counting of Distinct Elements in the Turnstile Model and Extensions

Monika Henzinger ✉ 

Institute of Science and Technology, Klosterneuburg, Austria

A. R. Sricharan ✉ 

Faculty of Computer Science, Doctoral School Computer Science, University of Vienna, Austria

Teresa Anna Steiner ✉ 

Technical University of Denmark, Lyngby, Denmark

---

## Abstract

---

Privately counting distinct elements in a stream is a fundamental data analysis problem with many applications in machine learning. In the turnstile model, Jain et al. [NeurIPS2023] initiated the study of this problem parameterized by the maximum flippancy of any element, i.e., the number of times that the count of an element changes from 0 to above 0 or vice versa. They give an item-level  $(\epsilon, \delta)$ -differentially private algorithm whose additive error is tight with respect to that parameterization. In this work, we show that a very simple algorithm based on the sparse vector technique achieves a tight additive error for item-level  $(\epsilon, \delta)$ -differential privacy and item-level  $\epsilon$ -differential privacy with regards to a different parameterization, namely the sum of all flippancies. Our second result is a bound which shows that for a large class of algorithms, including all existing differentially private algorithms for this problem, the lower bound from item-level differential privacy extends to event-level differential privacy. This partially answers an open question by Jain et al. [NeurIPS2023].

**2012 ACM Subject Classification** Security and privacy

**Keywords and phrases** differential privacy, turnstile model, counting distinct elements

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.40

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2408.11637>

**Funding** *Monika Henzinger:* This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (MoDynStruct, No. 101019564) and the Austrian Science Fund (FWF) grant DOI 10.55776/Z422, grant DOI 10.55776/I5982, and grant DOI 10.55776/P33775 with additional funding from the netidee SCIENCE Stiftung, 2020–2024.

*Teresa Anna Steiner:* Supported by a research grant (VIL51463) from VILLUM FONDEN.



## 1 Introduction

Counting distinct elements in a stream is a fundamental data analysis problem that is widely studied [12, 13, 17, 18, 20] and has many applications [1, 10, 19, 2, 21, 5], including network analysis [21] and detection of denial of service attacks [1, 5]. If the data includes sensitive information, the essential challenge is to give accurate answers while providing privacy guarantees to the data owners. Differential privacy is the de-facto standard in private data analysis and is widely employed both in research and in industry. In the insertions-only model, the problem of counting distinct elements while preserving differential privacy is well-studied [3, 9, 14].

Recent work by Jain, Kalemaj, Raskhodnikova, Sivakumar, and Smith [16] (which was concurrent with an earlier version of the results presented in this paper, see [15, Section 5]) initiated the study of this problem in the more general turnstile model. They give an algorithm which is *item-level*,  $(\epsilon, \delta)$ -differentially private and analyze the additive error



© Monika Henzinger, A. R. Sricharan, and Teresa Anna Steiner;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 40; pp. 40:1–40:21



Leibniz International Proceedings in Informatics

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

parameterized in the *maximum flippancy* of any element, i.e., the number of times that the count of an element changes from 0 to above 0 or vice versa. They also give lower bounds which show that the additive error of the algorithm is tight for item-level differential privacy (up to log factors) with respect to their parameterization. There is still a gap for event-level differential privacy, which is posed as an open question. The algorithm is based on several instantiations of the binary tree mechanism.

In this paper we show that a simple algorithm based on the sparse vector technique achieves a tight additive error (up to log factors) for item-level  $(\epsilon, \delta)$ -differential privacy and item-level  $\epsilon$ -differential privacy, with regards to a different parameterization, namely the *total flippancy*, i.e., the sum of the flippancies of all elements. The additive error depends polynomially on the total flippancy with a smaller exponent than the exponent of the maximum flippancy in the additive error in [16]. Thus, if there are few elements in total, or few elements which change their count from 0 to above 0 or vice versa, then our algorithm achieves a better additive error. Additionally, we give a reduction which shows that for a large class of algorithms, including all existing differentially private algorithms for this problem, the lower bound from item-level differential privacy extends to event-level differential privacy. This is a step towards answering the open question posed in [16].

## 1.1 Problem Definition

More formally, we assume there are  $d$  different items, and our goal is to maintain a multiset of them and to determine at each time step how many of them are currently at least once in the multiset, i.e., the number of distinct elements in the multiset. The update operations are modeled as follows: The input at every time step is a  $d$ -dimensional vector  $x^t \in \{-1, 0, 1\}^d$ , such that  $x_i^t = 1$  if element  $i$  gets inserted at time  $t$ ,  $x_i^t = -1$  if element  $i$  gets deleted at time  $t$ , and  $x_i^t = 0$  otherwise. Note that this means that we allow *multiple non-zero entries* in  $x^t$ , corresponding to *multiple updates* at every time step. However, the lower bound also extends to the case where we assume that at most one element may be inserted or deleted at any time step, i.e.,  $\|x^t\|_1 \leq 1$ , which we call *singleton update streams*. At every time step  $t$ , we want to output the number of distinct elements in the multiset. By our definition of the input stream, an element  $i$  is present at time  $t$  if and only if  $\sum_{t' \leq t} x_i^{t'} > 0$ .

► **Definition 1** (COUNTDISTINCT). *Let  $x^1, x^2, \dots, x^T$  be an input stream with  $x^t \in \{-1, 0, 1\}^d$  for all  $t \in [1, T]$ . We define  $\text{COUNTDISTINCT}(x)^t = \sum_{i=1}^d \mathbb{1}(\sum_{t' \leq t} x_i^{t'} > 0)$ , where  $\mathbb{1}(E)$  is the indicator function that is 1 if  $E$  is true and 0 otherwise. Then, the COUNTDISTINCT problem is to output  $\text{COUNTDISTINCT}(x)^t$  at all time steps  $t$ . The error of COUNTDISTINCT is defined to be the maximum additive error over all time steps.*

In this paper, we consider two privacy notions: *event-level* differential privacy, and *item-level* differential privacy. They differ in their definition of neighboring input streams. Two input streams  $x$  and  $y$  are *event-level neighboring*, if there exists a time step  $t^*$  and an item  $i^* \in [1, d]$  such that we have  $x_i^t = y_i^t$  for all  $(i, t) \neq (i^*, t^*)$ . That is, two event-level neighboring streams may differ in *at most one item in at most one update operation*. Two input streams  $x$  and  $y$  are *item-level neighboring*, if there exists an item  $i^* \in [1, d]$  such that  $x_i^t = y_i^t$  for all  $t$  and for all  $i \in [1, d] \setminus \{i^*\}$ . That is, two item-level neighboring streams may differ in *all update operations related to one item*.

Finally, we consider two models regarding the input stream. In the *general model* the counts for any item at any time step  $t$  is given by  $\sum_{t' \leq t} x_i^{t'}$ , which can be any integer in  $[-t, t]$  and we only care about whether  $\sum_{t' \leq t} x_i^{t'}$  is larger than zero or not. In the *“likes”-model*<sup>1</sup>

<sup>1</sup> The name was chosen as it models the count of “likes” on a social media website, as motivated by [16].

for every item  $i$  at any time step  $t$ , it must hold that  $\sum_{t' \leq t} x_i^{t'} \in \{0, 1\}$ , i.e., the multiset is a set. Said differently, an item can only be inserted if it is absent in the set and it can only be deleted when it is present.

## 1.2 Summary of Results

In this paper, we give new upper and lower bounds for *item-level* differential privacy, parameterized in the *total flippancy*  $K$ , which is defined as the total number of times any item switches from a non-zero count to a zero count, or vice versa. In detail, let  $f^t(x_i) = \mathbb{1}(\sum_{t' < t} x_i^{t'} > 0)$ . The total flippancy is formally defined as  $K = \sum_{i=1}^d \sum_{t=2}^T \mathbb{1}(f^t(x_i) \neq f^{t-1}(x_i))$ . Note that in the “likes”-model, the total flippancy is equal to the total number of updates. As  $\text{COUNTDISTINCT}(x)^t = \sum_{i=1}^d f^t(x_i)$ , it follows that  $K$  is an upper bound on the number of changes in  $\text{COUNTDISTINCT}(x)$  over time.

### Upper Bounds

As our first main result, we give algorithms solving  $\text{COUNTDISTINCT}$  while providing item-level differential privacy, which work in the general model (thus also in the “likes”-model). In the following, we state the exact bounds for *given*  $K$ . If  $K$  is not given to the algorithm, the error bounds worsen by at most a  $\ln^2 K$  factor.

► **Theorem 2.** *Let  $d$  be a non-zero integer,  $\beta > 0$ ,  $K$  be a known upper bound on the total flippancy, and let  $T$  be a known upper bound on the number of time steps. Then there exists*

1. *an item-level  $\epsilon$ -differentially private algorithm for  $\text{COUNTDISTINCT}$  in the general model with additive error  $O(\min(d, K, \sqrt{\epsilon^{-1} K \ln(T/\beta)}, \epsilon^{-1} T \log(T/\beta)))$  with probability at least  $1 - \beta$  at all time steps simultaneously, for any  $\epsilon > 0$  and  $\beta \in (0, 1)$ ;*
2. *an item-level  $(\epsilon, \delta)$ -differentially private algorithm for  $\text{COUNTDISTINCT}$  in the general model with additive error  $O(\min(d, K, (\epsilon^{-2} K \ln(1/\delta) \ln^2(T/\beta))^{1/3}, \epsilon^{-1} \sqrt{T \ln(1/\delta) \log(T/\beta)}))$  with probability at least  $1 - \beta$  at all time steps simultaneously, for any  $\delta \in (0, 1)$ ,  $\epsilon \in (0, 1)$ , and  $\beta \in (0, 1)$ .*

As our lower bounds (discussed below) show, our bounds for  $\epsilon$ -differential privacy are *tight*, if  $K$  is known and  $K \leq T$ . If  $K > T$ , we incur at most an extra  $\ln T$  factor, and if  $K$  is not known, we incur extra  $\ln K$  factors (see Section 6). For  $(\epsilon, \delta)$ -differential privacy, the upper bounds are tight up to  $\ln T$ ,  $\ln K$  and  $\ln(1/\delta)$  factors.

### Lower bounds

We complement our upper bounds by almost tight lower bounds on the additive error which hold for any item-level differentially private algorithm in the “likes”-model. As this is the “more restricted” of the two models, the lower bounds also carry over to the general model. For  $\epsilon$ -differential privacy, our lower bound follows from a packing argument.

► **Theorem 3** (Simplified version of Theorem 16). *For any  $L \leq T$ , there exists an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$ , which is valid in the “likes”-model, with length  $T$  and flippancy  $K = \Theta(L)$ , such that any item-level,  $\epsilon$ -differentially private algorithm for  $\text{COUNTDISTINCT}$  must with constant probability have an error at least  $\Omega(\min(d, K, \sqrt{\epsilon^{-1} K \max(\ln(T/K), 1)}))$ .*

The lower bound above also holds for singleton updates. When multiple updates are allowed, then  $K$  could potentially be larger than  $T$ . In that case, Theorem 16 in Section 5 shows that for any  $T \leq L \leq dT$ , there exists a stream with flippancy  $K = \Omega(T)$ ,

■ **Table 1** Comparison of our results (in blue) and the results in [16] for the different models.  $K$  denotes the total flippancy and  $w$  denotes the maximum flippancy of an input stream  $x$ . For simplicity of exposition, we consider singleton insertions and omit factors polynomial in  $\ln T$ ,  $\ln(1/\delta)$ , and  $\epsilon^{-1}$ . The bounds marked with \* hold for *output dependent* algorithms (see the discussion before Theorem 5 for details). The bounds in the last line follow from a simple application of a continual counting algorithm on the difference sequence.

|                            | Item-level $\epsilon$ -dp           | Item-level $(\epsilon, \delta)$ -dp                               | Event-level $\epsilon$ -dp                     | Event-level $(\epsilon, \delta)$ -dp                              |
|----------------------------|-------------------------------------|---|--|---|
| general model<br>[16]      | $\Omega(\min(w, \sqrt{T}))$         | $O(\min(\sqrt{w}, T^{1/3}))$<br>$\Omega(\min(\sqrt{w}, T^{1/3}))$ |  | $O(\min(\sqrt{w}, T^{1/3}))$<br>$\Omega(\min(\sqrt{w}, T^{1/4}))$ |
| “likes”-model<br>[16]      | $\Omega(\min(w, \sqrt{T}))$         | $O(\min(\sqrt{w}, T^{1/3}))$<br>$\Omega(\min(\sqrt{w}, T^{1/3}))$ |  | $O(\min(\sqrt{w}, T^{1/3}))$                                      |
| general model<br>this work | $O(\sqrt{K})$<br>$\Omega(\sqrt{K})$ | $O(K^{1/3})$<br>$\Omega(K^{1/3})$                                 | $O(\sqrt{K})$<br>$\Omega(\min(w, \sqrt{K}))^*$ | $O(K^{1/3})$<br>$\Omega(\min(\sqrt{w}, K^{1/3}))^*$               |
| “likes”-model<br>this work | $O(\sqrt{K})$<br>$\Omega(\sqrt{K})$ | $O(K^{1/3})$<br>$\Omega(K^{1/3})$                                 | $O(\sqrt{K})$                                  | $O(K^{1/3})$  |
| “likes”-model              |                                     |   | $O(1)$   | $O(1)$  |

$K = O(L)$ , such that any item-level,  $\epsilon$ -differentially private algorithm to the COUNT-DISTINCT problem must have error at least  $\Omega(\min(d, \epsilon^{-1}T, \sqrt{\epsilon^{-1}L \max(\ln(T/L), 1)})) = \Omega(\min(d, \epsilon^{-1}T, \sqrt{\epsilon^{-1}K \max(\ln(T/K), 1)}))$  with constant probability. For  $(\epsilon, \delta)$ -differential privacy, we can use a similar strategy as in [16] to get the following bounds:

► **Theorem 4** (Simplified version of Theorem 19). *Let  $\epsilon, \delta \in (0, 1]$ . Let  $K, T$  be sufficiently large parameters. There exists a dimension  $d \in \mathbb{N}$  and an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$  of length  $T$  with flippancy at most  $K$  which is valid in the “likes”-model, such that any item-level,  $(\epsilon, \delta)$ -differentially private algorithm for COUNTDISTINCT must have error at least  $\Omega\left(\epsilon^{-1} \cdot \min\left(\frac{\sqrt{T}}{\log T}, \frac{(K\epsilon)^{1/3}}{\log(K\epsilon)}\right)\right)$  with constant probability.*

Note that this lower bound holds for the case where *multiple insertions* are allowed in every time step. In Theorem 19 we also give a lower bound of  $\Omega\left(\frac{K^{1/3}}{\epsilon \log K}\right)$  for singleton-updates.

### Time and Space Complexity

Our main algorithm (achieving the  $O(\sqrt{\epsilon^{-1}K \ln T})$  error bound for  $\epsilon$ -differential privacy and  $O((K \ln(1/\delta) \ln^2 T)^{1/3}/\epsilon^{2/3})$  error bound for  $(\epsilon, \delta)$ -differential privacy) can be implemented using constant time per update, assuming that drawing from a Laplace distribution takes constant time. Specifically, the total running time is  $O(\#\text{updates} + S_K t_{\text{Lap}})$ , where  $t_{\text{Lap}}$  is the time to draw one Laplace random variable,  $S_K = O(\sqrt{K\epsilon/\ln T} + 1)$  for  $\epsilon$ -dp, and  $S_K = O\left(\left(\frac{K\epsilon}{\sqrt{\ln(1/\delta) \ln(T/\beta)}}\right)^{2/3}\right)$  for  $(\epsilon, \delta)$ -dp. The algorithm uses  $O(d)$  words of space. The only information our algorithm needs to store are the true counts for each item, plus a constant number of words of extra information. This holds even for the case where  $K$  is unknown, since we *sequentially* run our known  $K$  algorithm with increasing guesses for  $K$ .

### Comparison to the recent work by Jain, Kalemaj, Raskhodnikova, Sivakumar, and Smith [16]

In recent work, [16] considered the COUNTDISTINCT problem with a similar, but different parameterization. In [16], they parameterize the additive error in the *maximum flippancy*, i.e., they parameterize on  $w_x = \max_{i \in [d]} (\sum_{t=2}^T \mathbb{1}(f^t(x_i) \neq f^{t-1}(x_i)))$ . Recall that  $K$  denotes the

total flippancy of a stream  $x$  and note that  $w_x \leq K \leq d \cdot w_x$ . [16] consider only streams with singleton updates and give algorithms for item-level,  $(\epsilon, \delta)$ -differential privacy in the general model, with an error bound of  $\tilde{O}\left(\min\left(\left(\sqrt{w_x} \log T + \log^3 T\right) \cdot \frac{\sqrt{\log(1/\delta)}}{\epsilon}, \frac{(T \log(1/\delta))^{1/3}}{\epsilon^{2/3}}, T\right)\right)^2$ .

In comparison, our bounds in this setting are  $\tilde{O}\left(\min\left(\frac{(K \ln(1/\delta) \ln^2 T)^{1/3}}{\epsilon^{2/3}}, K\right)\right)$ . Note that  $K \leq T$  for singleton updates, and thus, our upper bounds recover their second and third bound up to a  $\ln^{2/3} T$  factor. Furthermore, ignoring polynomial factors in  $\log T$ ,  $\log(1/\delta)$  and  $\epsilon^{-1}$ , their bound is  $O(\sqrt{w_x})$  while ours is  $O(K^{1/3})$ . Thus, if (roughly)  $K < w_x^{3/2}$ , our algorithm outperforms theirs. Specifically, if  $d \leq \sqrt{w_x}$  or if there are only few items with high flippancy, we expect our algorithm to do better. In cases where the flippancy is well-distributed, i.e., many items have a similar flippancy, and  $d \geq \sqrt{w_x}$ , we expect the algorithm in [16] to perform better.

In terms of space and time complexity, their algorithm, like ours, needs to maintain a count for each element. Thus, the space in terms of words is  $\Omega(d)$ . On top of that, they run a variant of the binary tree mechanism, which depending on the implementation, uses  $\Omega(\log T)$  space. In their final solution, they actually run  $\log T$  copies of the binary tree mechanism in parallel, bringing their space consumption to  $O(d + \log^2 T)$  words. Thus, the space of our algorithm is an additive  $\log^2 T$  term better, which can be crucial for large streams. In terms of time complexity, each of the binary tree mechanism needs to draw  $\Omega(T \log T)$  independent Laplace noises, thus their time complexity is at least  $\Omega(T \log^2 T t_{\text{Lap}})$ , where  $t_{\text{Lap}}$  is the time it takes to draw a Laplace noise. Also here, our algorithm is more efficient.

In terms of lower bounds, for item-level,  $\epsilon$ -differential privacy in the “likes”-model, [16] give a lower bound of  $\Omega(\min(\epsilon^{-1}w, \sqrt{\epsilon^{-1}T}, T))$  for streams of maximum flippancy at most  $w$ . For  $(\epsilon, \delta)$ -differential privacy, they give a lower bound of  $\tilde{\Omega}(\min(\epsilon^{-1}\sqrt{w}, \epsilon^{-2/3}T^{1/3}, T))$  for item-level privacy in the “likes”-model, and  $\tilde{\Omega}(\min(\epsilon^{-1}\sqrt{w}, \epsilon^{-3/4}T^{1/4}, T))$  for event-level privacy in the general model, for streams of maximum flippancy at most  $w$ . Their upper bounds in the item-level setting match their lower bounds up to factors polynomial in  $\log T$  and  $\log(1/\delta)$ . For event-level in the general model, there is a gap for  $\sqrt{T} \leq w \leq T^{2/3}$ , and closing this gap was posed as an explicit open question in [16]. As our second main result, we make a step towards closing this gap, which we explain below.

### Reduction from item-level, “likes”-model to output-dependent event-level, general model

All upper bounds mentioned so far hold for *item-level* differential privacy. As our upper bounds hold in the general model and our lower bounds hold in the “likes”-model, we can conclude that for item-level privacy, the “likes”-model and the general model are roughly equally hard. [16] arrived at this conclusion as well, albeit with a different parameterization.

However, for *event-level* differential privacy, the picture is different: for the “likes”-model, a very simple algorithm gives an error of  $O(\epsilon^{-1} \text{polylog}(T))$  with constant probability. To see this, define the *difference sequence* for the COUNTDISTINCT problem as  $\text{diff}^t(x) = \text{COUNTDISTINCT}(x)^t - \text{COUNTDISTINCT}(x)^{t-1}$  for  $t > 1$ . As can be easily seen,  $(\text{diff}^t(x))_{t>1}$  and  $(\text{diff}^t(y))_{t>1}$  differ by at most 1 in at most one time step  $t$  for any event-level neighboring streams  $x$  and  $y$  in the “likes”-model. Thus, applying a standard continual counting algorithm gives the claimed error, as shown for “well-behaved” difference sequences in general in [11].

<sup>2</sup> For simplicity of exposition, we use  $\tilde{O}(X)$  to denote  $O(X \cdot \text{polylog}(X))$

For event-level differential privacy and the *general model* however, the best known algorithms are the algorithms for item-level differential privacy in this paper and [16]. [16] also present lower bounds for event-level differential privacy in the general model which, however, leave a gap for certain parameter settings. Closing that gap was explicitly posed as an open question in [16]. We make a step towards closing that gap, by noting that all existing differentially private algorithms for the COUNTDISTINCT problem in *any* model share the following property: If  $\text{COUNTDISTINCT}(x) = \text{COUNTDISTINCT}(y)$  for any two input streams  $x$  and  $y$ , then the output distributions of the algorithms are equal. That is, any two streams which produce the same true output, will have the same output distributions. We call such algorithms *output-determined*. We show that if we only consider output-determined algorithms for COUNTDISTINCT, then achieving *event-level differential privacy in the general model is just as hard as item-level differential privacy for the “likes”-model*. Thus our above lower bounds also apply to such algorithms. In particular, this shows that if one were trying to close the gap for event-level differential privacy in the general model, one needs to find an algorithm which does not only depend on the true answers to COUNTDISTINCT.

► **Theorem 5** (Simplified version of Theorem 15). *Let  $\epsilon > 0$  and  $\delta \geq 0$ . Let  $\mathcal{A}_1$  be an event-level,  $(\epsilon, \delta)$ -differentially private, output-determined algorithm for COUNTDISTINCT that works in the general model and has error at most  $\alpha$  for streams of length  $T + 1$  with probability  $1 - \beta$ . Then there exists an item-level,  $(2\epsilon, (1 + e^\epsilon)\delta)$ -differentially private algorithm  $\mathcal{A}_2$  for COUNTDISTINCT that works in the “likes”-model, and has error at most  $\alpha$  for streams of length  $T$  with probability  $1 - \beta$ .*

### Generalizations & Applications

While our algorithms are (nearly) tight for the COUNTDISTINCT problem, they are not tailored specifically to the problem and work in a more general setting as well. In particular, recall that  $\text{COUNTDISTINCT}(x)^t = \sum_{i=1}^d f^t(x_i)$ , where  $f^t(x_i) = \mathbb{1}(\sum_{t' \leq t} x_i^{t'} > 0)$ . Now consider any real-valued function  $Q$  on input streams  $x_1, x_2, \dots$ , with  $x_i \in \{-1, 0, 1\}$ . We use  $Q^t(x)$  to denote  $Q(x_1, \dots, x_t)$ . Our algorithm works for any such function  $Q$  such that the following two conditions are fulfilled: (1) for any  $x$  and  $y$  which are neighboring, we have  $|Q^t(x) - Q^t(y)| \leq 1$  for all time steps  $t$ , and (2)  $\sum_{t=1}^T |Q^t(x) - Q^{t-1}(x)| \leq K$ .

► **Theorem 6.** *Let  $Q$  be a function satisfying properties (1) and (2). Then there exists*

1. *an item-level  $\epsilon$ -differentially private algorithm for computing  $Q$  with additive error*

$$O(\min(K, \sqrt{\epsilon^{-1} K \ln(T/\beta)}), \epsilon^{-1} T \log(T/\beta))$$

*at all time steps with probability at least  $1 - \beta$ , for any  $\epsilon > 0$ ;*

2. *an item-level  $(\epsilon, \delta)$ -differentially private algorithm for computing  $Q$  with additive error*

$$O(\min(K, (\epsilon^{-2} K \ln(1/\delta) \ln^2(T/\beta))^{1/3}, \epsilon^{-1} \sqrt{T \ln(1/\delta) \log(T/\beta)}))$$

*at all time steps with probability at least  $1 - \beta$ , for any  $\epsilon > 0$ ,  $\delta \in (0, 1)$ ;*

The extension to unknown  $K$  also holds, with extra  $\ln K$  factors as earlier. Thus, for a continuous function  $Q$  which has *maximum* sensitivity 1 *over all time steps*, we get a bound parameterized in the sum of all differences, i.e., the  $L_1$ -norm of the difference sequence. While our results hold in the *turnstile model* and the additive error is parameterized by the total flippancy, [11] gave an  $\epsilon$ -differentially private mechanism with additive error  $O(\Gamma \log^{3/2} \log(T/\beta))$  in the *insertions-only* or *deletions-only* setting, where  $\Gamma$  is the *continuous global sensitivity* which is the  $L_1$ -norm of the difference sequence of two neighboring inputs.



We can apply our algorithm to the problem considered in Fichtenberger et al. [11] of continuously counting high degree nodes under differential privacy, which counts the number of nodes with degree at least  $\tau$ , where  $\tau$  is given and public. For user-level, edge-differential privacy (i.e., neighboring streams may differ in all updates of the same edge), they give a lower bound of  $\Omega(n)$ . Our algorithm gives new parameterized bounds for this problem: In particular, choosing  $Q^t(x) = \frac{\# \text{ of high degree nodes}}{2}$ , Theorem 6 gives an error bound of roughly  $O(\sqrt{K})$ , under  $\epsilon$ -differential privacy, and roughly  $O(K^{1/3})$  under  $(\epsilon, \delta)$ -differential privacy, where we ignore an  $O(\epsilon^{-1} \ln T \ln(1/\delta) \ln K)$  factor. Note that  $K$  can be as large as  $T$ , but for many applications, it could be much smaller: for example, in social networks, it has been shown that the degree distribution follows a power-law distribution, which implies that the set of high-degree nodes only changes infrequently.  $K$  does not be given to the algorithm.

### 1.3 Algorithm Overview

The main idea of our algorithm is to use the sparse vector technique first introduced by Dwork, Naor, Reingold, Rothblum, and Vadhan [7] (the form we use it in can be found in Dwork and Roth [8]) on carefully chosen queries and with carefully chosen thresholds. The sparse vector technique can be described as follows: It is given a data set  $x$ , a sequence of  $q$  queries, a threshold  $Thresh$ , and a stopping parameter  $S$ . It will process these queries sequentially, and for each of them answer “yes” or “no” depending on whether or not  $q(x)$  is approximately (up to an additive error  $\alpha$ ) above the threshold. It stops after it has answered “yes”  $S$  times. Dwork and Roth [8] show that it is possible to design an  $\epsilon$ -differentially private algorithm achieving the above with  $\alpha = O(\epsilon^{-1} S \log(q/\beta))$  with probability  $1 - \beta$ , and an  $(\epsilon, \delta)$ -differentially private algorithm with  $\alpha = O(\epsilon^{-1} \sqrt{S \log(1/\delta)} \log(q/\beta))$  with probability  $1 - \beta$ . In the following discussion, we ignore  $\epsilon^{-1}$ ,  $\log(1/\delta)$ ,  $\log q$  and  $\log(1/\beta)$  factors.

Our main idea is to note that the total flippancy  $K$  can be seen as an upper bound on *the total change in the output*, i.e., the sum of the absolute differences in the output in every time step. Our strategy is as follows: We start by estimating the number of distinct elements at the beginning of the stream. Then, we keep reporting this estimate until a significant change occurs in the true number of distinct elements. We track whether such a change has occurred using the sparse vector technique. Once there has been a significant change, i.e., once the sparse vector technique answers “yes”, we update the output. The goal now is to balance the additive error of the sparse vector technique with the error accumulated between updates. The error between updates is roughly  $Thresh$ ; the error of the sparse vector technique is  $\alpha$ ; and the total change of the output is bounded by  $K$ . To balance the two we set  $Thresh = \Theta(\alpha)$ . Furthermore we have to choose  $S$  in a way that makes sure that the sparse vector technique does not abort before we have seen the entire stream. We can show that every time our sparse vector technique answers “yes”, the change in output has been roughly  $Thresh$ . Thus it is enough to set  $S > K/Thresh$ . As mentioned above, for  $\epsilon$ -differential privacy  $\alpha$  (and, thus,  $Thresh$ ) must depend linearly on  $S$ , which implies that  $S$  must be chosen to be  $\Theta(\sqrt{K})$ , giving an additive error of  $O(\sqrt{K})$ . For  $(\epsilon, \delta)$ -differential privacy, we have  $Thresh = \Theta(\alpha) = O(\sqrt{S})$ . This implies that  $S^{3/2}$  must be  $\Theta(K)$ , i.e.,  $S = \Theta(K^{2/3})$ . Thus the additive error is  $O(K^{1/3})$ .

Note that this requires that  $K$  is known at the beginning of the algorithm. If  $K$  is unknown, we run the above algorithm for exponentially increasing guesses of  $K$  ( $K = 2, 4, 8$ , etc.). In particular, we run the algorithm for a guess of  $K$ , and if it terminates preemptively, we double our guess and repeat. Since we do not know beforehand how many instances are needed, in order to make sure the resulting algorithm is still  $\epsilon$ -differentially private, we run the  $j$ th instance with privacy parameter  $\epsilon_j = O(\epsilon/j^2)$ , such that  $\sum_{j=1}^{\infty} \epsilon_j \leq \epsilon$ . At the end of the algorithm,  $j = \Theta(\ln K)$ , therefore we incur an extra  $\ln^2 K$  factor in the additive error.

## 2 Preliminaries

We denote  $\{1, \dots, n\}$  by  $[n]$  and the input stream length by  $T$ , the number of time steps.

### Continual observation algorithm

An algorithm  $A$  in the continual observation model gets an update at every time step  $t \leq T$ , and produces an output  $a^t = A(x^1, \dots, x^t)$  which is a function of  $x^1$  to  $x^t$ ;  $A^T(x) = (a^1, a^2, \dots, a^T)$  denotes the sequence of outputs at all time steps.

► **Definition 7** (Differential privacy [6]). *A randomized algorithm  $A$  is  $(\epsilon, \delta)$ -differentially private  $((\epsilon, \delta)$ -dp) if for all  $S \in \text{range}(A^T)$  and all  $x, y$  neighboring*

$$\Pr[A^T(x) \in S] \leq e^\epsilon \Pr[A^T(y) \in S] + \delta.$$

If  $\delta = 0$  then  $A$  is  $\epsilon$ -differentially private ( $\epsilon$ -dp).

► **Definition 8** (Laplace Distribution). *The Laplace distribution centered at 0 with scale  $b$  is the distribution with probability density function  $f_{\text{Lap}}(b)(x) = (2b)^{-1} \cdot \exp(-|x|/b)$ . We use  $X \sim \text{Lap}(b)$  or just  $\text{Lap}(b)$  to denote a random variable  $X$  distributed according to  $f_{\text{Lap}}(b)(x)$ .*

In our definitions below, we use  $\chi$  to represent a generic universe of elements.

► **Definition 9** (Sensitivity). *Let  $f : \chi \rightarrow \mathbb{R}^k$ . The  $L_p$ -sensitivity  $\Delta_p$  is defined as*

$$\max_{x \in \chi, y \in \chi, x \sim y} \|f(x) - f(y)\|_p,$$

where  $x \sim y$  denotes that  $x$  and  $y$  are neighbouring.

► **Fact 1** (Theorem 3.6 in [8]: Laplace Mechanism). *Let  $f$  be any function  $f : \chi \rightarrow \mathbb{R}^k$  with  $L_1$ -sensitivity  $\Delta_1$ . Let  $Y_i \sim \text{Lap}(\Delta_1/\epsilon)$  for  $i \in [k]$ . The mechanism defined as  $A(x) = f(x) + (Y_1, \dots, Y_k)$  satisfies  $\epsilon$ -differential privacy.*

► **Fact 2** (Laplace Tailbound). *If  $X \sim \text{Lap}(b)$ , then  $\Pr[|X| \geq t \cdot b] \leq e^{-t}$ .*

The following fact follows from Theorem A.1 in [8]:

► **Fact 3** (Gaussian Mechanism). *Let  $f$  be any function  $f : \chi \rightarrow \mathbb{R}^k$  with  $L_2$ -sensitivity  $\Delta_2$ . Let  $Y_i \sim \mathcal{N}(0, \sigma^2)$  for  $i \in [k]$ , where  $\sigma \geq \sqrt{2 \ln(2/\delta)} \Delta_2/\epsilon$ . The mechanism defined as  $A(x) = f(x) + (Y_1, \dots, Y_k)$  satisfies  $(\epsilon, \delta)$ -differential privacy.*

► **Fact 4** (Gaussian tailbound). *If  $X \sim \mathcal{N}(0, \sigma^2)$ , then  $\Pr[|X| \geq \sigma \sqrt{\ln(2/\beta)}] \leq \beta$*

The following facts are respectively given by Theorem 3.16, 3.20 and Corollary 3.21 in [8].

► **Fact 5** (Composition Theorem). *Let  $A_1$  be an  $(\epsilon_1, \delta_1)$ -differentially private algorithm  $A_1 : \chi \rightarrow \text{range}(A_1)$  and  $A_2$  an  $(\epsilon_2, \delta_2)$ -differentially private algorithm  $A_2 : \chi \times \text{range}(A_1) \rightarrow \text{range}(A_2)$ . Then  $B : \chi \rightarrow \text{range}(A_1) \times \text{range}(A_2)$  defined as  $B(x) = (A_1(x), A_2(x, A_1(x)))$  is  $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ -differentially private.*

► **Fact 6** (Advanced Composition). *Let  $\epsilon, \delta, \delta' \geq 0$ . Let  $A_1$  be an  $(\epsilon, \delta)$ -differentially private algorithm  $A_1 : \chi \rightarrow \text{range}(A_1)$  and  $A_i$  be  $(\epsilon, \delta)$ -differentially private algorithms  $A_i : \chi \times \text{range}(A_{i-1}) \rightarrow \text{range}(A_i)$ , for  $2 \leq i \leq k$ . Then the composition  $B : \chi \rightarrow \text{range}(A_1) \times \dots \times \text{range}(A_k)$  defined as  $B(x) = (A_1(x), A_2(x, A_1(x)), \dots, A_k(x, A_{k-1}(x)))$  is  $(\epsilon', k\delta + \delta')$ -differentially private, where  $\epsilon' = \sqrt{2k \ln(1/\delta')} \epsilon + k\epsilon(e^\epsilon - 1)$ .*

► **Corollary 10**. *Let  $\epsilon^*, \delta, \delta' \geq 0$  and  $\delta', \epsilon^* < 1$ . Let  $A_1, \dots, A_k$  be as in Fact 6 with*

$$\epsilon = \epsilon^*/(2\sqrt{2k \ln(1/\delta')}).$$

*Then the composition  $B$  (defined as in Fact 6) is  $(\epsilon^*, k\delta + \delta')$ -differentially private.*

### 3 Item-Level Algorithms in General Model

In this section, we give algorithms which work for any input sequence in the general model, and thus also for input sequences that fulfill the conditions of the “likes”-model. The upper bounds on the additive error for  $\epsilon$ -differential privacy match the lower bounds in Section 5, except for the  $\log(T/\beta)$  factor in the case where  $K > T$ .

#### 3.1 Known Total Flippancy

We prove Theorem 11 in this section. We give some intuition first on Algorithm 1. The algorithm works by iteratively checking if the true number of distinct elements currently present (called  $Q$ ) is “far” from the current output of our algorithm (called  $out$ ) using a sparse vector technique (SVT) instantiation. We start the algorithm by estimating  $out$  at the beginning of the stream (line 8). Then, we keep outputting  $out$ , while we track the difference between  $out$  and the true number of distinct elements  $Q$  (line 14). Once there has been a significant change, we update the output (line 18).

There are two parameters of interest here. One is the number of times we update the output: we abort after  $S_K$  updates happen (line 21). The other is the parameter  $Thresh$ , which determines how large the current error needs to be such that we satisfy the condition in line 14. The parameter  $S_K$  goes into the error from composition, while the parameter  $Thresh$  directly goes into the additive error bound.

The goal is to balance the error accumulated between updates (which is roughly  $Thresh$ ), and the error from updating  $out$  privately (which is roughly  $S_K$  for  $\epsilon$ -differential privacy, and roughly  $\sqrt{S_K}$  for  $(\epsilon, \delta)$ -differential privacy due to composition). Additionally, we want to make sure our algorithm does not abort before having processed the entire stream. We show that every time SVT returns “yes”, the total flippancy in the stream has increased by at least  $\Omega(Thresh)$ . Since we know the total flippancy is bounded by  $K$ , in order to make sure that we do not abort preemptively, we choose  $S_K$  such that  $S_K \cdot Thresh \approx K$ . Balancing the two error terms yields an additive error of approximately  $\sqrt{K}$  for  $\epsilon$ -differential privacy, and  $K^{1/3}$  for  $(\epsilon, \delta)$ -differential privacy.

► **Theorem 11.** *Let  $d$  and  $T$  be non-zero integers, let  $\beta > 0$ , and let  $K$  be an upper bound on the total flippancy which is given. Let  $T$  be a known upper bound on the number of time steps. Then there exists*

1. *an item-level  $\epsilon$ -differentially private algorithm for COUNTDISTINCT in the general model with error at most  $O(\min(d, K, \sqrt{\epsilon^{-1}K \ln(T/\beta)}, \epsilon^{-1}T \log(T/\beta)))$  at all time steps with probability at least  $1 - \beta$  for  $\epsilon > 0$ ;*
2. *an item-level  $(\epsilon, \delta)$ -differentially private algorithm for COUNTDISTINCT in the general model with error  $O\left(\min(d, K, (\epsilon^{-2}K \ln(1/\delta) \ln^2(T/\beta))^{1/3}, \epsilon^{-1}\sqrt{T \ln(1/\delta) \log(T/\beta)})\right)$  at all time steps with probability at least  $1 - \beta$ , for any  $0 < \delta < 1$  and  $0 < \epsilon < 1$ .*

**Proof.** The  $O(\min(d, K))$  bound follows from the fact that the algorithm that outputs 0 at every time step is  $\epsilon$ -differentially private and has error at most  $\min(d, K)$  for any  $\epsilon$ . The third error bounds in the minimum for Theorem 11 are achieved by Algorithm 1, as shown below. Since we assume here all parameters are known, one can compute the minimum of the three bounds and choose the algorithm accordingly. The fourth bound in Theorem 11 follow by a direct application of the Laplace mechanism Fact 1 with  $\Delta_1 = T$  resp. Gaussian mechanism Fact 3 with  $\Delta_2 = \sqrt{T}$ .

■ **Algorithm 1** COUNTDISTINCT, known  $K$ .

---

```

1: Input: Data Stream  $x = x^1, x^2, \dots$ , initial counts  $c_1, \dots, c_d$  (default 0), parameters  $\epsilon, \delta$ 
   and  $\beta$ , stream length bound  $T$ , stopping parameter  $S_K \geq 1$ 
2: if  $\delta = 0$  then  $\epsilon_1 = \epsilon/(2S_K)$ 
3: if  $\delta > 0$  then  $\epsilon_1 = \epsilon/(4\sqrt{2S_K \ln(1/\delta)})$ 
4: count = 1
5:  $\tau_1 = \text{Lap}(2/\epsilon_1)$ 
6:  $\nu_1 = \text{Lap}(1/\epsilon_1)$ 
7:  $Q = 0$ 
8: out =  $Q + \nu_1$ 
9: Thresh =  $16\epsilon_1^{-1}(\ln(2T/\beta))$ 
10: for  $t = 1, \dots$ , do
11:    $c_i = c_i + x_i^t$  for all  $i \in [d]$ 
12:    $Q = |\{i \in [d] \mid c_i > 0\}|$ 
13:    $\mu_t = \text{Lap}(4/\epsilon_1)$ 
14:   if  $|\text{out} - Q| + \mu_t > \text{Thresh} + \tau_{\text{count}}$  then
15:     count = count + 1
16:      $\tau_{\text{count}} = \text{Lap}(2/\epsilon_1)$ 
17:      $\nu_{\text{count}} = \text{Lap}(1/\epsilon_1)$ 
18:     out =  $Q + \nu_{\text{count}}$ 
19:   end if
20:   output out
21:   if count  $\geq S_K$  then Abort
22: end for

```

---

The algorithm for our third bound, given in Algorithm 1, is based on the sparse vector technique, where  $S_K$  is a parameter dependent on  $K$  that we choose suitably below. We omit the proof of the following lemma, since it follows from well-known techniques (Sparse Vector Technique [7, 8], Laplace mechanism (Fact 1) and composition theorems (Facts 5 and 6)).

► **Lemma 12.** *For  $\delta = 0$  and any  $\epsilon > 0$ , Algorithm 1 is  $\epsilon$ -differentially private. For  $0 < \epsilon < 1$  and  $0 < \delta < 1$ , Algorithm 1 is  $(\epsilon, \delta)$ -differentially private.*

We show the claimed accuracy bound using the following lemma.

► **Lemma 13.** *For  $\delta = 0$ , for any time step  $t$  before the algorithm aborts, we have that the maximum error up to time  $t$  is at most  $O(\epsilon^{-1}S_K \ln(T/\beta))$ . Setting  $S_K = \sqrt{K\epsilon/(18\ln(T/\beta))} + 1$ , with probability at least  $1 - \beta$ , Algorithm 1 does not abort before having seen the entire stream, and has error at most  $O(\sqrt{\epsilon^{-1}K \ln(T/\beta)} + \epsilon^{-1} \ln(T/\beta))$ . For  $\delta > 0$ , for any time step  $t$  before the algorithm aborts, we have that the maximum error up to time  $t$  is  $O(\epsilon^{-1}\sqrt{S_K \ln(1/\delta)} \ln(T/\beta))$ . Setting  $S_K = \left(\frac{K\epsilon}{36\sqrt{\ln(1/\delta)} \ln(T/\beta)}\right)^{2/3} + 1$ , with probability at least  $1 - \beta$ , Algorithm 1 does not abort before having seen the entire stream, and has error at most  $O\left(\left(\epsilon^{-2}K \ln(1/\delta) \ln^2(T/\beta)\right)^{1/3} + \epsilon^{-1}\sqrt{\ln(1/\delta)} \ln(T/\beta)\right)$ .*

**Proof.** Note that at every time step  $t$  in Algorithm 1, we set  $Q = \sum_{i=1}^d f^t(x_i)$ . Let  $\alpha = (8/\epsilon_1) \ln(2T/\beta) = (1/2) \cdot \text{Thresh}$ . By Laplace tailbounds (Fact 2), at every time step  $t$ :  
**(a)**  $|\tau_\ell| \leq (2/\epsilon_1) \ln(2T/\beta) = \alpha/4$  with probability at least  $1 - \beta/(2T)$ , where  $\ell$  is the value of variable count at time step  $t$ , and

(b)  $|\mu_\ell| \leq (4/\epsilon_1) \ln(2T/\beta) = \alpha/2$  with probability at least  $1 - \beta/(2T)$ .

Thus, with probability  $\geq 1 - \beta$ , we have at all time steps  $t$  simultaneously:

- (i) Whenever the condition in line 14 is true at time  $t$ , then  $|\text{out} - \sum_{i \in [d]} f^t(x_i)| > \text{Thresh} - 3\alpha/4 = 5\alpha/4$ , and
- (ii) Whenever the condition in line 14 is false at time  $t$ , then  $|\text{out} - \sum_{i \in [d]} f^t(x_i)| \leq \text{Thresh} + 3\alpha/4 < 3\alpha$ .

Further, the random variable  $\nu_\ell$  for  $\ell \in [S_K]$  is distributed as  $\text{Lap}(1/\epsilon_1)$  and is added to  $\sum_{i \in [d]} f^t(x_i)$  at every time step  $t$  where out is updated. By the Laplace tail bound (Fact 2),  $\nu_\ell$  is bounded for all  $\ell \in [S_K]$  by  $\epsilon_1^{-1} \ln(S_K/\beta) \leq \alpha/8$  with probability at least  $1 - \beta$ . Altogether, all of these bounds hold simultaneously with probability at least  $1 - 2\beta$ . We condition on all these bounds being true.

Assume the algorithm has not terminated yet at time  $t$  and let out be the value of variable out at the beginning of time  $t$ . Let  $p_\ell$  be the last time step at which the value of out was updated. It holds that  $|\text{out} - \sum_{i \in [d]} f_i^{p_\ell}(x)| = |\nu_\ell| \leq \alpha/8$ . If the condition in line 14 is true at time  $t$ , then

$$\left| \sum_{i \in [d]} f_i^{p_\ell}(x) - \sum_{i \in [d]} f^t(x_i) \right| \geq \left| \sum_{i \in [d]} f^t(x_i) - \text{out} \right| - \left| \text{out} - \sum_{i \in [d]} f_i^{p_\ell}(x) \right| \geq 5\alpha/4 - \alpha/8 = 9\alpha/8.$$

Thus, between two time steps where the value of out is updated, there is a change of at least  $9\alpha/8$  in the sum value, i.e., the value of  $f^t(x_i)$  has changed at least once for  $\geq 9\alpha/8$  different items  $i$ . Since  $K = \sum_{i=1}^d \sum_{t=2}^T \mathbb{1}(f^t(x_i) \neq f^{t-1}(x_i))$ , to guarantee (under the noise conditions), that the algorithm does not terminate before we have seen the entire stream, it suffices to choose  $S_K$  where  $S_K > K/(9\alpha/8)$ .

For  $\delta = 0$ , we have  $\alpha = (8/\epsilon_1) \ln(2T/\beta) = (16S_K/\epsilon) \ln(2T/\beta)$ , thus we have to choose  $S_K > K\epsilon/(18S_K \ln(2T/\beta))$ . Choosing  $S_K = \lfloor \sqrt{K\epsilon/(18 \ln(2T/\beta))} \rfloor + 1$  fulfills this condition.

Similarly, for  $\delta > 0$ , choosing  $S_K = \left( \frac{K\epsilon}{36\sqrt{\ln(1/\delta)} \ln(T/\beta)} \right)^{2/3} + 1$  fulfills this condition.

Now consider any time step  $t$  and let out be the output at time  $t$ , i.e., the value after processing time step  $t$ . If the condition in line 14 is false, we showed above that  $|\text{out} - \sum_{i \in [d]} f^t(x_i)| < 3\alpha$ . If the condition is true at time  $t$ , we have  $\text{out} = \sum_{i \in [d]} f^t(x_i) + \nu_\ell$  for some  $\ell \in [S_K]$ , and, thus,  $|\text{out} - \sum_{i \in [d]} f^t(x_i)| \leq \alpha/8 < \alpha$ .

For  $\delta = 0$ , we have  $\alpha = (8/\epsilon_1) \ln(2T/\beta) = O(\sqrt{\epsilon^{-1}K \ln(T/\beta)} \ln(T/\beta) + \epsilon^{-1} \ln(T/\beta))$ .

Plugging in  $S_K = \left( \frac{K\epsilon}{36\sqrt{\ln(1/\delta)} \ln(T/\beta)} \right)^{2/3} + 1$  yields the final bound for  $\delta > 0$ . ◀

To finish the proof of Theorem 11, note that if  $\epsilon^{-1} \ln(T/\beta) > \sqrt{\epsilon^{-1}K \ln(T/\beta)}$ , then  $\sqrt{\epsilon^{-1}K \ln(T/\beta)} > K$ , which can be seen by multiplying both sides of the inequality with  $\sqrt{K}/\sqrt{\epsilon^{-1} \ln(T/\beta)}$ . Thus the upper bound  $\min(d, K, \sqrt{\epsilon^{-1}K \ln(T/\beta)})$  holds for  $\delta = 0$ .

Also, if  $\epsilon^{-1} \sqrt{\ln(1/\delta)} \ln(T/\beta) > (\epsilon^{-2}K \ln(1/\delta) \ln^2(T/\beta))^{1/3}$ , then  $\epsilon^{-1} \sqrt{\ln(1/\delta)} \ln(T/\beta) > K$ , which can be seen by first cubing the inequality and then dividing by  $\epsilon^{-2} \ln(1/\delta) \ln^2(T/\beta)$ . Thus, for  $\delta > 0$ , the upper bound of  $\min(d, K, (\epsilon^{-2}K \ln(1/\delta) \ln^2(T/\beta))^{1/3})$  holds. ◀

### 3.2 Generalizations

We now argue about Theorem 6. Let  $Q$  be a real-valued function on input streams from  $\{-1, 0, 1\}$  and let  $Q^t = Q(x_1, \dots, x_t)$ . Further, let  $Q$  be such that 1.) for any  $x$  and  $y$  which are neighboring, we have  $|Q^t(x) - Q^t(y)| \leq 1$  for all time steps  $t$ , and 2.)  $\sum_{t=1}^T |Q^t(x) - Q^{t-1}(x)| \leq$

$K$ . The first bound from Theorem 6 is achieved by an algorithm that never updates the output, and the third bounds for  $\epsilon$  and  $(\epsilon, \delta)$ -differential privacy are obtained by the Laplace and Gaussian mechanisms, respectively. The second bound for both  $\epsilon$  and  $(\epsilon, \delta)$ -differential privacy is obtained by Algorithm 1 by setting  $Q = Q^t(x)$  at every time step  $t$ . The proofs follow by exchanging  $\sum_{i \in [d]} f^t(x_i)$  by  $Q^t(x)$  in the proofs of Lemma 12 and 13.

#### 4 A Connection between the General Model under Event-Level Privacy and the “Likes”-Model under Item-Level Privacy

Our bounds from Theorems 2, 3, and 4 as well as the bounds from [16] imply that under *item-level privacy*, the “likes”-model and the general model are roughly equally hard: all upper bounds hold for the general model and all lower bounds hold for the “likes”-model, and the bounds are tight up to a  $\log T$  factor. However, under *event-level privacy*, the “likes”-model is significantly easier than the general model: It can be solved via continual counting on the difference sequence of the true output, which gives error polylogarithmic in  $\log T$ . This is possible because for event-level privacy in the “likes”-model, the difference sequence of the output (i.e., the difference between the true output value of the current and the preceding time step) has  $\ell_\infty$ -sensitivity 1 for event-level privacy, but for item-level privacy, the sensitivity can be as large as  $T$ .

In the general model, there are no better upper bounds known for event-level differential privacy than for item-level differential privacy, and the upper and lower bounds from [16] for  $(\epsilon, \delta)$ -differential privacy for the event-level setting in the general model leave a polynomial (in  $T$ ) gap, in the case where the maximum flippancy  $w_x \in (T^{1/2}, T^{2/3})$ : In that case, ignoring polynomial factors in  $\epsilon^{-1}$ ,  $\log(1/\delta)$ , and  $\log T$ , the lower bound of [16] is  $\Omega(T^{1/4})$ , while their algorithm gives an additive error of  $O(T^{1/3})$ . Specifically, finding the best achievable error for *event-level privacy* in the general model is explicitly posed as an open question in [16].

We resolve this question for a large class of algorithms, called  $\gamma$ -*output-determined* algorithms. All known algorithms for this problem in *any* model are 0-output-determined. Specifically, we show that for  $\gamma$ -output-determined algorithms our lower bounds and the lower bounds from [16] for *item-level privacy* in the “likes”-model basically carry over to *event-level privacy* in the *general model*. It follows that our algorithm and the algorithm from [16] for event-level privacy in the general model are tight up to a factor that is linear in  $\log T$  *within the class of output-determined algorithms*. Note that our reduction works both for the  $\epsilon$ -differential privacy as well as for  $(\epsilon, \delta)$ -differential privacy and we give the corresponding lower bounds in Theorems 16 and 19. In the following, we denote by  $\text{COUNTDISTINCT}(x)$  the stream of true answers to the  $\text{COUNTDISTINCT}$  problem on stream  $x$ .

► **Definition 14.** Let  $\gamma \geq 0$ . An algorithm  $\mathcal{A}$  for the  $\text{COUNTDISTINCT}$  problem is said to be  $\gamma$ -output-determined, if for all inputs  $x$  and  $y$  such that  $\text{COUNTDISTINCT}(x) = \text{COUNTDISTINCT}(y)$  and any  $S \in \text{range}(\mathcal{A})$  we have:

$$\Pr(\mathcal{A}(x) \in S) \leq \Pr(\mathcal{A}(y) \in S) + \gamma$$

► **Theorem 15.** Let  $\epsilon > 0, \delta \geq 0$  and  $\gamma \geq 0$ . Let  $\mathcal{A}_1$  be an event-level,  $(\epsilon, \delta)$ -differentially private,  $\gamma$ -output-determined algorithm for  $\text{COUNTDISTINCT}$  that works in the general model and has error at most  $\alpha$  for streams of length  $T+1$  with probability  $1 - \beta$ . Then there exists an item-level,  $(2\epsilon, (1 + e^\epsilon)\delta + e^\epsilon\gamma)$ -differentially private algorithm  $\mathcal{A}_2$  for  $\text{COUNTDISTINCT}$  that works in the “likes”-model, and has error at most  $\alpha$  for streams of length  $T$  with probability  $1 - \beta$ .

**Proof.** We describe algorithm  $\mathcal{A}_2$ , that is item-level  $(2\epsilon, (1 + e^\epsilon)\delta + e^\epsilon\gamma)$ -dp in the “likes”-model, derived from a  $\gamma$ -output-determined algorithm  $\mathcal{A}_1$  which is event-level,  $(\epsilon, \delta)$ -dp in the general model: Let  $x$  be an input for COUNTDISTINCT in the “likes”-model of length  $T$ , i.e.,  $x$  is such that  $\sum_{t' \leq t} x_i^{t'}$  can only take the values 0 or 1, for any  $i \in [d]$  and  $t \in [T]$ . Let  $x_0 = 0^d x$ , i.e., we attach a  $d$ -dimensional all-zero vector before  $x$ , and define  $(\mathcal{A}_2(x))^t = (\mathcal{A}_1(x_0))^{t+1}$  for all  $t \in [T]$  (note that  $\mathcal{A}_1$  can take inputs from the “likes”-model). We now show that  $\mathcal{A}_2$  is item-level  $(2\epsilon, (1 + e^\epsilon)\delta + e^\epsilon\gamma)$ -differentially private. Let  $x$  and  $y$  be two item-level neighbouring inputs in the “likes”-model. That is, there exists an item  $i$  such that the streams  $x_i$  and  $y_i$  may be completely different, while  $x_j = y_j$  for all  $j \neq i$ . Additionally, since we are in the “likes”-model, for any time step  $t$ ,  $\sum_{t' \leq t} x_i^{t'} \in \{0, 1\}$  and  $\sum_{t' \leq t} y_i^{t'} \in \{0, 1\}$ .

Next, we define input streams  $z$  and  $w$  in the general model where  $\text{COUNTDISTINCT}(z) = \text{COUNTDISTINCT}(w)$ ,  $z$  is event-level neighbouring to  $x_0$ , and  $w$  is event-level neighbouring to  $y_0$ . Since  $\mathcal{A}_1$  is event-level  $(\epsilon, \delta)$ -dp and works for the general model, we then have for any  $S \in \text{range}(\mathcal{A}_2)$

$$\begin{aligned} \Pr[\mathcal{A}_2(x) \in S] &= \Pr[(\mathcal{A}_1(x_0))_{t=2}^{T+1} \in S] \leq e^\epsilon \Pr[(\mathcal{A}_1(z))_{t=2}^{T+1} \in S] + \delta \\ &\leq e^\epsilon \Pr[(\mathcal{A}_1(w))_{t=2}^{T+1} \in S] + \delta + \gamma \\ &\leq e^{2\epsilon} \Pr[(\mathcal{A}_1(y_0))_{t=2}^{T+1} \in S] + (1 + e^\epsilon)\delta + e^\epsilon\gamma \\ &= e^{2\epsilon} \Pr[\mathcal{A}_2(y) \in S] + (1 + e^\epsilon)\delta + e^\epsilon\gamma, \end{aligned}$$

where the second inequality holds as  $\mathcal{A}_1$  is  $\gamma$ -output-determined.

To define such  $z$  and  $w$ , let  $-e_i$  be the vector such that  $-e_i(j) = 0$  for all  $j \neq i$  and  $-e_i(i) = -1$ . Then  $z = -e_i x$  and  $w = -e_i y$ . Note that  $z$  and  $w$  are valid input streams for the general model, while they are not valid for the “likes”-model. Clearly,  $z$  is event-level neighbouring to  $x_0$ , and  $w$  is event level neighbouring to  $y$ . Recall that  $\text{COUNTDISTINCT}(z)^t = \sum_{j=1}^d \mathbb{1}(\sum_{t' \leq t} z_j^{t'} > 0)$ . Since  $\sum_{t' \leq t} x_i^{t'} \in \{0, 1\}$  for all  $t \in [T]$  we have  $\sum_{t' \leq t} z_i^{t'} \leq 0$  for all  $t \in [T + 1]$ . By the same argument, we have  $\sum_{t' \leq t} w_i^{t'} \leq 0$  for all  $t \in [T + 1]$ . Since  $z$  and  $w$  only differ in the  $i$ th coordinate, which never contributes to the COUNTDISTINCT value as it is never 1, we have  $\text{COUNTDISTINCT}(z) = \text{COUNTDISTINCT}(w)$ .

We are left with analyzing the error of the two algorithms. For this, note that by definition of  $x_0$ , we have  $\text{COUNTDISTINCT}(x_0)^{t+1} = \text{COUNTDISTINCT}(x)^t$ . Thus, running  $\mathcal{A}_2$  on  $x$  gives the same error as running  $\mathcal{A}_1$  on  $x_0$ . ◀

In particular, for any output-determined algorithm, Theorem 15 implies that all lower bounds on the error for the COUNTDISTINCT problem under *item-level* differential privacy which hold for the “likes”-model (and thus, all lower bounds for COUNTDISTINCT under item-level differential privacy shown in this paper in Theorem 19 and in [16]), carry over to *event-level* differential privacy in the *general model*. This means that if there is an algorithm achieving a better error than the bounds stated in Theorem 19 and in [16] for event-level differential privacy in the general model, it cannot be  $\gamma$ -output-determined for  $\gamma = O(\delta)$ , i.e., it must be such that it does not *only* depend on the number of distinct elements at any given time step.

## 5 Item-Level Lower Bounds in the “Likes”-Model

In the following we show lower bounds for solving COUNTDISTINCT under item-level differential privacy, and in the “likes”-model. The lower bounds also apply to the general model. In Section 3, we showed a complementing upper bound which holds in the general model, even if  $K$  is unknown to the algorithm.

► **Theorem 16.** *Let  $d$  and  $T > 4$  be non-negative integers and let  $\epsilon > 0$ .*

1. *Let  $L \geq 8$  be a non-negative integer such that  $L \leq dT$ . There exists an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$ , which is valid in the “likes”-model with multiple updates per time step, with length  $T$  and flippancy  $K$  with  $\min(3L/8, T/4 - 1) \leq K \leq \min(L, dT/4)$  such that any  $\epsilon$ -differentially private algorithm to the COUNTDISTINCT problem with item-level privacy with error at most  $\alpha$  at all time steps with probability at least  $2/3$  must satisfy*

$$\begin{aligned} \alpha &= \Omega(\min(d, L, \epsilon^{-1}T, \sqrt{\epsilon^{-1}L \max(\ln(T/L), 1)})) \\ &= \Omega(\min(d, K, \epsilon^{-1}T, \sqrt{\epsilon^{-1}K \max(\ln(T/K), 1)}). \end{aligned}$$

2. *Let  $L \geq 8$  be a non-negative integer such that  $L \leq T$ . There exists an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$ , which is valid in the “likes”-model with multiple updates per time step, with length  $T$ , flippancy  $K$  with  $L/16 \leq K \leq \min(L, T/4)$ , and with  $\|x^t\|_1 = 1$  for all  $t$  (i.e., each update modifies at most one item) such that any  $\epsilon$ -differentially private algorithm to the COUNTDISTINCT problem with item-level privacy with error at most  $\alpha$  at all time steps with probability at least  $2/3$  must satisfy*

$$\alpha = \Omega(\min(d, K, \sqrt{\epsilon^{-1}K \ln(T/K)}).$$

**Proof.** Let  $d$ ,  $T$ , and  $L$  be as given in the theorem statement. Assume there is an  $\epsilon$ -differentially private algorithm  $\mathcal{A}$  for the COUNTDISTINCT problem with error at most  $\alpha$  at all time steps with probability at least  $2/3$ . If  $\alpha > d/2$ , then the error is  $\Omega(d)$ . Also, if  $\alpha > L/8$ , then  $\alpha = \Omega(L)$ . Thus, in the following, we consider the case  $\alpha \leq d/2$  and  $\alpha \leq L/8$ . Defining  $m = \lfloor 2\alpha \rfloor$ , it follows that  $m \leq \min(d, L/8)$ .

**Singleton updates.** We first find  $T' \leq T$  and  $L' \leq L$  such that  $4m$  divides  $T'$  and  $m$  divides  $L'$ . If this is not the case for  $T$  and  $L$ , then pick parameters  $T'$  and  $L'$  such that (i)  $4m$  divides  $T'$  and  $m$  divides  $L'$ , (ii)  $\Delta = T - T' \leq 4m < L/2 \leq T/2$  (i.e.  $T' = \Theta(T)$ ) and (iii)  $0 \leq L - \Delta - L' \leq m$ . This implies that  $L' \geq 7L/8 - \Delta \geq 3L/8$ . Thus, as  $L \leq T$ , then  $0 \leq L - \Delta - L' = L - (T - T') - L' = T' - L' - (T - L) \leq T' - L'$ , i.e.,  $L' \leq T'$ .

We use  $T'$  and  $L'$  in the proof below to construct a sequence of length  $T'$  fulfilling the statements of the theorem. To complete the proof of the theorem, we append to the sequence  $T - T'$  many all-zero vectors to guarantee that the stream has length  $T$ . Note that appending to the sequence “blank” operation will not invalidate the statements of the theorem.

We now construct a set of input sequences of length  $T'$  with flippancy  $K := \min(L', T'/4)$  and use them to prove a lower bound for  $\alpha$  of  $\Omega(\min(K \ln(T'/K), \sqrt{\epsilon^{-1}K \ln(T'/K)}))$ . Combined with the above case distinctions giving lower bounds on  $\alpha$  of  $\Omega(d)$ , and  $\Omega(L)$ , the fact that  $K = \Theta(L)$  and that  $T' = \Theta(T)$ , this implies that  $\alpha = \Omega(\min(d, K, \sqrt{\epsilon^{-1}K (\ln(T'/K) + 1)}))$ .

Let  $k := \min(L', T'/4)/m$  be a positive integer. Partition the timeline into  $T'/m$  blocks of length  $m$ , namely  $B_1 = [1, m]$ ,  $B_2 = [m + 1, 2m]$ ,  $\dots$ . Now, for any subset of blocks  $J = (j_1, \dots, j_k)$  with  $1 \leq j_1 < j_2 < \dots < j_k \leq T'/m$ , define an input sequence  $x(J)$  such that for any item  $i \in [m]$  we insert element  $i$  in the  $i$ th time step of every odd block of  $J$  (i.e. the first, third,  $\dots$  block in  $J$ ), and delete it again at the  $i$ th position of every even block of  $J$  (i.e. the second, fourth,  $\dots$  block in  $J$ ). More formally, for any item  $i \in [m]$ , set  $x(J)_i^t = 1$  for all  $t = B_{j_{2p-1}}[i] = (j_{2p-1} - 1)m + i$ ,  $p = 1 \dots, \lfloor k/2 \rfloor$ , and set  $x(J)_i^t = -1$  for all  $t = B_{j_{2p}}[i] = (j_{2p} - 1)m + i$ ,  $p = 1 \dots, \lfloor k/2 \rfloor$ . In all other time steps  $t$ , no updates are performed, i.e.,  $x(J)^t$  is an all-zero vector. Thus, for every  $i \in [m]$ , we have  $f^t(x_i) = 1$



for all time steps  $t \in [j_{2p-1}m, (j_{2p} - 1)m]$ , for all  $p \leq \lceil k/2 \rceil$ , and  $f^t(x_i) = 0$  for all time steps  $t \in [j_{2p}m, (j_{2p+1} - 1)m]$ . For any item  $m < i \leq d$ , we have  $f^t(x_i) = 0$  for all  $t \in [T']$ . Furthermore, items  $i$  with  $i > m$  (if they exist) are never inserted or deleted. In total, there are  $k = \min(L', T'/4)/m$  updates per item  $i \in [m]$ , thus exactly  $K$  updates in total, and, hence, the total flippancy is  $K = \min(L', T'/4)$ . If  $K = L'$ , then  $L \geq K \geq 3L/8$ . If  $K = T'/4$ , then  $L' \leq T'$  implies that  $L \geq L' \geq K = T'/4 \geq L'/4 \geq 3L/32 \geq L/16$ . Thus in either case  $K = \Theta(L')$ . Furthermore  $K \leq T'/4 \leq T/4$ .

Now let  $E_J$ , for  $J = (j_1, \dots, j_k)$  with  $1 \leq j_1 < j_2 < \dots < j_k \leq T'/m$ , be the set of output sequences where  $\mathcal{A}$  outputs (i) a value of  $m/2$  or larger for all time steps  $t \in [j_{2p-1}m, (j_{2p} - 1)m]$  with  $1 \leq p \leq \lceil k/2 \rceil$ , and (ii) smaller than  $m/2$  for all time steps  $t$  such that (a)  $t < j_1m$  or (b)  $t \in [j_{2p}m, (j_{2p+1} - 1)m]$  for some  $0 \leq p < \lceil k/2 \rceil$  or (c)  $t \geq j_km$ . Note that for an input sequence  $x(J)$  every output sequence where  $\mathcal{A}$  has additive error smaller than  $\alpha = m/2$  must belong to  $E_J$ . As the algorithm is correct with probability at least  $2/3$ ,  $\Pr[\mathcal{A}(x(J)) \in E_J] \geq 2/3$ .

Two input sequences are neighboring if they differ in the data of at most one item for item-level differential privacy. As two input sequences  $x(I)$  and  $x(J)$  with  $I \neq J$  differ in the data of at most  $m$  items, it follows by group privacy that  $\Pr[\mathcal{A}(x(J)) \in E_I] \geq e^{-m\epsilon}2/3$  for any  $J = (j_1, \dots, j_k)$  with  $1 \leq j_1 < j_2 < \dots < j_k \leq T'/m$  and  $I = (i_1, \dots, i_k)$  with  $1 \leq i_1 < i_2 < \dots < i_k \leq T'/m$ . Also note that the set of output sequences  $E_J$  for distinct  $J = (j_1, \dots, j_k)$  are disjoint, since for each multiple of  $m$  (i.e., the end of a block), it is clearly defined whether the output is at least  $m/2$  or smaller than  $m/2$ , and as such the values  $j_1, \dots, j_k$  can be uniquely recovered. Thus, there are  $\binom{T'/m}{k}$  disjoint events  $E_J$  and the sum over all  $J$  of the probabilities that the algorithm with input  $x(I)$  outputs an event  $E_J$  is at most 1. More formally, we have:

$$1 \geq \binom{T'/m}{k} e^{-m\epsilon}2/3 \geq \frac{(T'/m)^k}{(k)^k} e^{-m\epsilon}2/3 = \frac{T'^{(K/m)}}{K^{K/m}} e^{-m\epsilon}2/3$$

where the last equality is since  $k = K/m$ . This gives

$$m^2 + \epsilon^{-1}m \ln(3/2) \geq \epsilon^{-1}K \ln(T'/K)$$

which implies

$$m = \Omega(\min(K \ln(T'/K), \sqrt{\epsilon^{-1}K \ln(T'/K)}).$$

Note that since  $T' \geq 4K$ ,  $\ln(T'/K) \geq \ln(4) > 1$ . This completes the proof.

**Multiple updates.** We first find  $T' \leq T$  and  $L' \leq L$  such that 4 divides  $T'$  and  $m$  divides  $L'$ . If this is not the case for  $T$  and  $L$ , then pick parameters  $T'$  and  $L'$  such that (i) 4 divides  $T'$  and  $m$  divides  $L'$ , (ii)  $\Delta = T - T' \leq 4$  (i.e.  $T' = \Theta(T)$ ) and (iii)  $\Delta m \leq L - L' \leq (\Delta + 1)m$ . This implies that  $L' \geq L - (\Delta + 1)m \geq L - 5m \geq 3L/8$ .

We use  $T'$  and  $L'$  in the proof below to construct a sequence of length  $T'$  fulfilling the statements of the theorem. To complete the proof of the theorem, we append to the sequence  $T - T'$  many all-zero vectors to guarantee that the stream has length  $T$ . Note that appending to the sequence “blank” operation will not invalidate the statements of the theorem.

The idea is similar to above, only we do not define blocks, but directly choose  $k := \min(L'/m, T'/4)$  time steps in which all items in  $[m]$  are updated. Thus the flippancy  $K$  will equal  $mk$ . More precisely, we construct the following set of input sequences. For any  $I = (t_1, \dots, t_k)$  with  $1 \leq t_1 < t_2 < \dots < t_k \leq T'$ , we define an input sequence  $x(I)$  as follows: For any item  $i \in [m]$ , set  $x(I)_i^{t_j} = 1$  for all odd  $j$ , and  $x(I)_i^{t_j} = -1$  for all even  $j$ .

## 40:16 Private Counting of Distinct Elements in the Turnstile Model and Extensions

All other coordinates are set to 0. In total, there are  $k$  updates per item in  $[m]$ , thus, exactly  $K$  updates in total, i.e., the total flippancy equals  $K = \min(L', mT'/4)$ . This implies that  $\min(3L/8, T/4 - 1) \leq K \leq \min(L, dT/4)$ .

Now, let  $E_I$ , for  $I = (t_1, \dots, t_k)$  with  $1 \leq t_1 < t_2 < \dots < t_k \leq T'$ , be the set of output sequences with a value of  $m/2$  or larger at all time steps  $t \in [t_{2p-1}, t_{2p})$  for some  $1 \leq p \leq \lceil k/2 \rceil$ , and a value smaller than  $m/2$  at all time steps  $t$  where (a)  $t \leq t_1$  or (b)  $t \in [t_{2p}, t_{2p+1})$  for some  $0 \leq p < \lceil k/2 \rceil$ . Note that for input sequence  $x(I)$  every output sequence where  $\mathcal{A}$  has an additive error smaller than  $m/2$  must be in  $E_I$ . As the algorithm is correct with probability at least  $2/3$ ,  $\Pr[\mathcal{A}(x(I)) \in E_I] \geq 2/3$ . As two input sequences  $x(I)$  and  $x(J)$  with  $I \neq J = (j_1, \dots, j_k)$  with  $1 \leq j_1 < j_2 < \dots < j_k \leq T'$  differ in the data of at most  $m$  items, it follows by group privacy that  $\Pr[\mathcal{A}(x(I)) \in E_J] \geq e^{-m\epsilon} 2/3$  for any such  $J$ .

Let  $J = (j_1, \dots, j_k)$  with  $1 \leq j_1 < j_2 < \dots < j_k \leq T'$ . Note that the events  $E_I$  and  $E_J$  for any  $I \neq J$  are disjoint, since in the event  $E_I$  it is clearly defined for every time step whether the output is at least  $m/2$  or smaller than  $m/2$ , and from that the set  $I$  can be uniquely recovered. Thus, there are  $\binom{T'}{k}$  disjoint events  $E_J$  and the probability that with input  $x(I)$  the algorithm outputs any one of them is at most 1. Thus we have

$$1 \geq \binom{T'}{k} e^{-m\epsilon} 2/3 \geq \frac{T'^k}{k^k} e^{-m\epsilon} 2/3 = \frac{T'^{K/m}}{(K/m)^{K/m}} e^{-m\epsilon} 2/3 \quad (1)$$

where the last equality is since  $k = K/m$ .

Next we consider two cases, the first one resulting in two different lower bounds on  $m$  and the second one giving a third lower bound on  $m$ . The combination of these three lower bounds then gives the claimed bound above of

$$\alpha = m/2 = \Omega(\min(\epsilon^{-1}T', \sqrt{\epsilon^{-1}K \max(\ln(T'/K), 1)}, K \max(\ln(T'/K), 1)))$$

**Case 1.**  $L' < mT'/4$ . In this case  $K = L'$  and we have

$$m^2\epsilon + m \ln(3/2) \geq K \ln(T'm/K) \geq K \max(\ln(T'/K), 1)$$

where the last inequality holds since  $K \leq mT'/4$ , i.e.,  $\ln(T'm/K) \geq \ln(4) > 1$ . Hence

$$m = \Omega(\min(\sqrt{\epsilon^{-1}K \max(\ln(T'/K), 1)}, K \max(\ln(T'/K), 1))).$$

As  $K = L' = \Theta(L)$  it follows that

$$m = \Omega(\min(\sqrt{\epsilon^{-1}L \max(\ln(T'/L), 1)}, L \max(\ln(T'/L), 1))).$$

**Case 2.**  $L' \geq mT'/4$ . This implies that  $K = mT'/4$  and, thus, that there are updates in at least  $T'/4$  many time steps. In this case Inequality 1 can be reformulated as follows:

$$1 \geq \frac{T'^{K/m}}{K/m^{K/m}} e^{-m\epsilon} 2/3 = 4^{T'/4} e^{-m\epsilon} 2/3 = e^{\ln(4)T'/4 - m\epsilon} 2/3,$$

which implies that Inequality 1 is satisfied for  $m = \Omega(\epsilon^{-1}T')$ .

These two cases show  $\alpha = \Omega(\min(\epsilon^{-1}T', \sqrt{\epsilon^{-1}L \max(\ln(T'/L), 1)}, L \max(\ln(T'/L), 1)))$  for the above input sequence. Combined with the above lower bounds on  $\alpha$  of  $\Omega(\min(d, L))$  and the fact that  $T' = \Theta(T)$ , it follows that  $\alpha = \Omega(\min(d, L, \epsilon^{-1}T, \sqrt{\epsilon^{-1}L \max(\ln(T/L), 1)}))$ . ◀

■ **Algorithm 2** COUNTDISTINCT, unknown  $K$ .

---

```

1: Input: Data Set  $D = x^1, x^2, \dots$ , initial counts  $c_1, \dots, c_d$  (default 0), parameters  $\epsilon, \delta$ 
   and  $\beta, T$ 
2:  $t = 1$ 
3:  $K_1 = 2$ 
4: for  $j = 1, \dots$ , do
5:    $\epsilon_j = 6\epsilon/(\pi^2 j^2)$ 
6:    $\delta_j = 6\delta/(\pi^2 j^2)$ 
7:    $\beta_j = 6\beta/(\pi^2 j^2)$ 
8:   if  $\delta = 0$  then
9:      $S_{K_j} = \sqrt{K_j \epsilon_j / (18 \ln(T/\beta_j))} + 1$ 
10:  end if
11:  if  $\delta > 0$  then
12:     $S_{K_j} = \left( \frac{K_j \epsilon_j}{36 \sqrt{\ln(1/\delta_j)} \ln(T/\beta_j)} \right)^{2/3} + 1$ 
13:  end if
14:  Run Algorithm 1 on  $x^t, x^{t+1}, \dots, c_1, \dots, c_d, \epsilon_j, \delta_j, \beta_j, T, S_{K_j}$  until it aborts
15:  Let  $t'$  be the last time step processed by Algorithm 1
16:   $t = t' + 1$ 
17:   $j = j + 1$ 
18:   $K_j = 2^j$ 
19: end for

```

---

## 6 Unknown Total Flippancy

The algorithms from Section 3 can be easily extended to the case where the total flippancy  $K$  is not known beforehand, at the cost of  $\text{polylog}(K)$  factors in the error bound, as shown by Algorithm 2 and the lemmata below. The fact that  $K$  is not known causes no serious problem, as the algorithm repeatedly “guesses”  $K$  and then runs the algorithm from earlier with the current guess.

► **Lemma 17.** *For any  $0 < \epsilon < 1$  and  $0 \leq \delta < 1$ , Algorithm 2 is  $(\epsilon, \delta)$ -differentially private.*

**Proof.** By Lemma 12, the  $j$ th instance of Algorithm 1 is  $(\epsilon_j, \delta_j)$ -differentially private. Since  $\sum_{j=1}^{\infty} \epsilon_j = \epsilon$  and  $\sum_{j=1}^{\infty} \delta_j = \delta$ , by Fact 5, Algorithm 2 is  $(\epsilon, \delta)$ -differentially private. ◀

► **Lemma 18.** *For  $\delta = 0$ , the error of Algorithm 2 is at most*

$$O(\ln K \sqrt{\epsilon^{-1} K \ln(T \ln K/\beta)} + \epsilon^{-1} \ln^2 K \ln(T \ln K/\beta)).$$

*For  $\delta > 0$ , the error of Algorithm 2 is at most*

$$O((\epsilon^{-1} K \ln^2 K \ln(\ln K/\delta) \ln^2(T \ln K/\beta))^{1/3} + \epsilon^{-1} \ln^2 K \sqrt{\ln(\ln K/\delta)} \ln(T \ln K/\beta)).$$

**Proof.** Let  $j_l$  be the value of variable  $j$  after the last element in the stream is processed. For any  $j < j_l$ , note that by Lemma 13, with probability at least  $1 - \beta_j$ , by the choice of  $S_{K_j}$ , the algorithm does not abort before having seen the entire stream if the total flippancy is at most  $K_j$ . Thus, when the algorithm aborts for some  $j < j_l$ , we know that the flippancy is at least  $K_j$ , and the bound from Lemma 13 holds for the  $j$ th instance of Algorithm 1 with  $S_{K_j}$ .

Since the algorithm aborts for all  $j < j_l$ , we can conclude that the total flippancy of the stream processed by the  $j$ th run of Algorithm 1 is at least  $K_j$ . Since  $\sum_j \beta_j = \beta$ , with probability at least  $1 - \beta$ , (1) the total flippancy  $K$  is at least  $\sum_{j < j_l} K_j = 2^{j_l} - 1$ , and

(2) the bound from Lemma 13 holds for all instances of Algorithm 1 (with their respective parameters). It follows (a) that  $K \geq K_{j_i} - 1 \geq K_j$  for all  $j < j_i$  and (b)  $j_i = O(\ln K)$ . The maximum error over the stream is the maximum error of any instance of Algorithm 1. Since  $K_j = O(K)$ ,  $\epsilon_{j_i} \leq \epsilon_j$  and  $\delta_{j_i} \leq \delta_j$  for all  $j \leq j_i$ , the final bound is now obtained by plugging  $K$ ,  $\epsilon_{j_i} = \Theta(\epsilon/j^2)$  for  $\epsilon$ ,  $\delta_{j_i} = \Theta(\delta/j^2)$  for  $\delta$ , and  $\beta_{j_i} = \Theta(\beta/j^2)$  for  $\beta$  into the bound from Lemma 13, and upper bounding  $j^2$  by  $\log^2 K$ . ◀

One can also obtain the minimum of the bound from Lemma 18 and  $\min(K, T, d)$  at the cost of an additive  $\epsilon \ln^2 K \ln(\ln K/\beta)$  factor with a slightly more involved algorithm, which involves choosing to either not update the output or abort if there is a trivial algorithm which performs better for the current estimate of  $K$ . If we knew the value of  $K$  beforehand, we could choose the best algorithm upfront. Not knowing the value of  $K$  makes it slightly more complicated. However, the algorithm and analysis are fairly straightforward, and we defer it to the full version.

## 7 Lower Bounds for Approximate Differential Privacy

In this section, we adapt the lower bounds from [16] for item-level differential privacy to our parameter scheme.

► **Theorem 19.** *Let  $\epsilon, \delta \in (0, 1]$ .*

1. *Let  $K, T$  be sufficiently large parameters. There exists a dimension  $d$  and an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$  of length  $T$  and with flippancy at most  $K$  which is valid in the “likes”-model, such that any item-level,  $(\epsilon, \delta)$ -differentially private algorithm to the COUNTDISTINCT problem with error at most  $\alpha$  at all time steps with probability at least 0.99 must satisfy  $\alpha = \Omega\left(\min\left(\frac{\sqrt{T}}{\epsilon \log T}, \frac{(K\epsilon)^{1/3}}{\epsilon \log(K\epsilon)}\right)\right)$ .*
2. *Let  $K$  and  $T$  be sufficiently large parameters satisfying  $K \leq T$ . There exists a dimension  $d$  and an input stream  $x$  of  $d$ -dimensional vectors from  $\{-1, 0, 1\}^d$  of length  $T$  and with flippancy at most  $K$  which is valid in the “likes”-model and satisfies  $\|x^t\|_1 = 1$  for all  $t$ , such that any item-level,  $(\epsilon, \delta)$ -differentially private algorithm to the COUNTDISTINCT problem with error at most  $\alpha$  at all time steps with probability at least 0.99 must satisfy  $\alpha = \Omega\left(\frac{K^{1/3}}{\epsilon \log K}\right)$ .*

The reduction in [16] is based on a lower bound for the 1-way marginals problem. In that problem, the data set  $y$  is a table consisting of  $n$  rows and  $m$  columns, where every entry is in  $\{0, 1\}$ . Two data sets  $y$  and  $y'$  are neighbouring if they differ in at most one row. The goal is to estimate the average column sums, i.e., the vector  $(\sum_{i=1}^n y[i, j])_{j \in [m]}$ . The following lower bound holds for estimating 1-way marginals under  $(\epsilon, \delta)$ -differential privacy:

► **Lemma 20** (Bun, Ullman, and Vadhan [4]). *Let  $\epsilon \in (0, 1]$ ,  $\gamma \in (0, 1)$ , and  $m, n \in \mathbb{N}$ , and  $\delta = o(1/n)$ . Any algorithm which is  $(\epsilon, \delta)$ -differential private and has error at most  $\gamma$  with probability at least 0.99 satisfies  $n = \Omega\left(\frac{\sqrt{m}}{\gamma \epsilon \log m}\right)$ .*

**Proof Sketch of Theorem 19.** We start by arguing about Item 2. For this case, our example stream is exactly the same as in [16], given in Algorithm 5 in [16] (for a formulation using our slightly different notation see Algorithm 3). They give a reduction from the 1-way marginals problem: For any instance  $\mathcal{I}$  of the 1-way marginals problem with  $n$  rows and  $m$  columns, there is an instance  $C(\mathcal{I})$  of COUNTDISTINCT with  $T = 2mn$ , such that if  $\mathcal{I}$  and  $\mathcal{I}'$  are neighbouring, then  $C(\mathcal{I})$  and  $C(\mathcal{I}')$  are item-neighbouring. Further, if we can solve  $C(\mathcal{I})$  within error  $\alpha$ , we can solve  $\mathcal{I}$  within error  $\alpha/n$ . It follows by Lemma 20 that  $\alpha = \Omega\left(\min\left(\frac{\sqrt{m}}{\epsilon \log m}, n\right)\right)$ . In the instance they constructed,  $d = n$ , i.e. each row in the 1-way

marginals problem gives an item in the COUNTDISTINCT problem. Further, the total flippancy  $K$  can be as large as  $2mn$  for worst case inputs. Thus, in order to apply the reduction, we need  $2mn \leq K \leq T$ . Given parameters  $K \leq T$ , we choose  $m = K/(2n)$ . The lower bound  $\Omega\left(\min\left(\frac{\sqrt{m}}{\epsilon \log m}, n\right)\right)$  translates to  $\Omega\left(\min\left(\frac{\sqrt{K/(2n)}}{\epsilon \log(K/(2n))}, n\right)\right)$ . For  $n = \frac{K^{1/3}}{2(\epsilon \log K)^{2/3}}$ , we have

$$\frac{\sqrt{K/(2n)}}{\epsilon \log(K/(2n))} \geq \frac{K^{1/3} \epsilon^{1/3} \log^{1/3} K}{\epsilon \log(K^{1/2})} = \Omega\left(\frac{K^{1/3} \log^{1/3} K}{\epsilon^{2/3} \log K}\right) = \Omega(n).$$

Thus, we get  $\alpha = \Omega\left(\frac{K^{1/3} \log^{1/3} K}{\epsilon^{2/3} \log K}\right)$ .

For Item 1., where we allow general updates, we have to slightly modify the example in [16]: namely, in their Algorithm 5, we collapse every one of their vectors  $z^{(j)}$ ,  $j = 1, \dots, m$ , into vectors of length 2, one time step for all insertions corresponding to column  $j$ , and one time step for all deletions corresponding to column  $j$ . See Algorithm 4. We then again get a reduction with the same properties as before, except that  $T = 2n$  and  $K$  can be as large as  $2mn$ . Now, the analysis from [16] can be repeated with our  $T$  taking the role of  $w_x$  in [16], and our  $K$  taking the role of  $T$  in [16]. ◀

■ **Algorithm 3** Algorithm 5 from [16]: Reduction from 1-way marginals to COUNTDISTINCT.

---

```

1: Input: Data Set  $y[1], \dots, y[n] \in \{0, 1\}^{n \times m}$  and blackbox access to a mechanism  $M$  for
   COUNTDISTINCT
2: Output: Estimates of marginals  $b = (b[1], \dots, b[m])$ 
3: for  $j = 1, \dots, m$  do
4:   for  $i = 1, \dots, n$  do
5:     Set  $z^{(j)}[i] = e_i$ 
6:     Set  $z^{(j)}[i + n] = -e_i$ 
7:   end for
8: end for
9: Run  $M$  on  $x \rightarrow z^{(1)} \circ z^{(2)} \circ \dots \circ z^{(m)}$  and record answer vector  $r$ 
10: for  $j \in [m]$  do do
11:    $b[j] = r[(2j - 1)n]/n$ 
12: end for
13: output  $b$ 

```

---

■ **Algorithm 4** Reduction from 1-way marginals to COUNTDISTINCT for arbitrarily many updates per round.

---

```

1: Input: Data Set  $y[1], \dots, y[n] \in \{0, 1\}^{n \times m}$  and blackbox access to a mechanism  $M$  for
   COUNTDISTINCT
2: Output: Estimates of marginals  $b = (b[1], \dots, b[m])$ 
3: for  $j = 1, \dots, m$  do
4:   Set  $z^{(j)}[1] = y^T[j]$ 
5:   Set  $z^{(j)}[2] = -y^T[j]$ 
6: end for
7: Run  $M$  on  $x \rightarrow z^{(1)} \circ z^{(2)} \circ \dots \circ z^{(m)}$  and record answer vector  $r$ 
8: for  $j \in [m]$  do do
9:    $b[j] = r[(2j - 1)]/n$ 
10: end for
11: output  $b$ 

```

---

---

**References**

---

- 1 Aditya Akella, Ashwin Barambe, Mike Reiter, and Srinivasan Seshan. Detecting ddos attacks on isp networks. In *Proceedings of the Workshop on Management and Processing of Data Streams*, pages 1–2, 2003.
- 2 Daniel N Baker and Ben Langmead. Dashing: fast and accurate genomic distances with hyperloglog. *Genome biology*, 20:1–12, 2019.
- 3 Jean Bolot, Nadia Fawaz, S. Muthukrishnan, Aleksandar Nikolov, and Nina Taft. Private decayed predicate sums on streams. In *Proc. 16th ICDT*, pages 284–295, 2013. doi:10.1145/2448496.2448530.
- 4 Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018. doi:10.1137/15M1033587.
- 5 Vera Clemens, Lars-Christian Schulz, Marten Gartner, and David Hausheer. Ddos detection in P4 using HYPERLOGLOG and COUNTMIN sketches. In *Proc. NOMS 2023*, pages 1–6, 2023.
- 6 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi:10.1007/11681878\_14.
- 7 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proc. 41st STOC*, pages 381–390, 2009. doi:10.1145/1536414.1536467.
- 8 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
- 9 Alessandro Epasto, Jieming Mao, Andres Muñoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. In Yael Tauman Kalai, editor, *Proc. 14th ITCS*, pages 48:1–48:24, 2023. doi:10.4230/LIPIcs.ITCS.2023.48.
- 10 Cristian Estan, George Varghese, and Michael E. Fisk. Bitmap algorithms for counting active flows on high-speed links. *IEEE/ACM Trans. Netw.*, 14(5):925–937, 2006.
- 11 Hendrik Fichtenberger, Monika Henzinger, and Lara Ost. Differentially private algorithms for graphs under continual observation. In *Proc. 29th ESA*, pages 42:1–42:16, 2021.
- 12 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.
- 13 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Proc. 2007 Conference on Analysis of Algorithms*, pages 127–146, 2007.
- 14 Badih Ghazi, Ravi Kumar, Jelani Nelson, and Pasin Manurangsi. Private counting of distinct and k-occurring items in time windows. In *Proc. 14th ITCS*, pages 55:1–55:24, 2023. doi:10.4230/LIPIcs.ITCS.2023.55.
- 15 Monika Henzinger, A. R. Sricharan, and Teresa Anna Steiner. Differentially private histogram, predecessor, and set cardinality under continual observation, 2023. arXiv:2306.10428.
- 16 Palak Jain, Iden Kalemaj, Sofya Raskhodnikova, Satchit Sivakumar, and Adam Smith. Counting distinct elements in the turnstile model with differential privacy under continual observation, 2023. arXiv:2306.06723.
- 17 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proc. 29th PODS*, pages 41–52, 2010.
- 18 Matti Karppa and Rasmus Pagh. Hyperlogloglog: Cardinality estimation with one log more. In *Proc. 28th KDD*, pages 753–761, 2022.

- 19 Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Why go logarithmic if we can go linear?: Towards effective distinct counting of search traffic. In *Proc. 11th EDBT*, pages 618–629, 2008.
- 20 Dingyu Wang and Seth Pettie. Better cardinality estimators for hyperloglog, pcsa, and beyond. In *Proc. 42nd PODS*, pages 317–327, 2023.
- 21 Lotte Weedage, Nelly Litvak, and Clara Stegehuis. Locating highly connected clusters in large networks with hyperloglog counters. *J. Complex Networks*, 9(2), 2021.






# Hilbert Functions and Low-Degree Randomness Extractors

Alexander Golovnev ✉ 🏠 

Georgetown University, Washington, DC, United States of America

Zeyu Guo ✉ 🏠 

The Ohio State University, Columbus, OH, United States of America

Pooya Hatami ✉ 🏠 

The Ohio State University, Columbus, OH, United States of America

Satyajeet Nagargoje ✉ 🏠 

Georgetown University, Washington, DC, United States of America

Chao Yan ✉ 🏠 

Georgetown University, Washington, DC, United States of America

---

## Abstract

For  $S \subseteq \mathbb{F}^n$ , consider the linear space of restrictions of degree- $d$  polynomials to  $S$ . The Hilbert function of  $S$ , denoted  $h_S(d, \mathbb{F})$ , is the dimension of this space. We obtain a tight lower bound on the smallest value of the Hilbert function of subsets  $S$  of arbitrary finite grids in  $\mathbb{F}^n$  with a fixed size  $|S|$ . We achieve this by proving that this value coincides with a combinatorial quantity, namely the smallest number of low Hamming weight points in a down-closed set of size  $|S|$ .

Understanding the smallest values of Hilbert functions is closely related to the study of degree- $d$  closure of sets, a notion introduced by Nie and Wang (Journal of Combinatorial Theory, Series A, 2015). We use bounds on the Hilbert function to obtain a tight bound on the size of degree- $d$  closures of subsets of  $\mathbb{F}_q^n$ , which answers a question posed by Doron, Ta-Shma, and Tell (Computational Complexity, 2022).

We use the bounds on the Hilbert function and degree- $d$  closure of sets to prove that a random low-degree polynomial is an extractor for samplable randomness sources. Most notably, we prove the existence of low-degree extractors and dispersers for sources generated by constant-degree polynomials and polynomial-size circuits. Until recently, even the existence of arbitrary deterministic extractors for such sources was not known.

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

**Keywords and phrases** Extractors, Dispersers, Circuits, Hilbert Function, Randomness, Low Degree Polynomials

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.41

**Category** RANDOM

**Related Version** *Full Version*: <https://eccc.weizmann.ac.il/report/2024/092/> [20]

**Funding** *Alexander Golovnev*: Supported by the NSF CAREER award (grant CCF-2338730)

*Pooya Hatami*: Supported by NSF grant CCF-1947546.

*Satyajeet Nagargoje*: Supported by the NSF CAREER award (grant CCF-2338730)

*Chao Yan*: Research supported in part by a gift to Georgetown University.

**Acknowledgements** We thank Omar Alrabiah, Jesse Goodman, Jonathan Mosheiff, and João Ribeiro for sharing with us an early draft of their work. We would also like to thank Jesse Goodman and S. Venkitesh for helpful discussions and pointers. We are very grateful to the anonymous reviewers for their comments and pointers to related work. Part of this work was conducted while the second author was visiting the Simons Institute for the Theory of Computing at UC Berkeley; he extends his thanks to the institute for its support and hospitality.



© Alexander Golovnev, Zeyu Guo, Pooya Hatami, Satyajeet Nagargoje, and Chao Yan; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 41; pp. 41:1–41:24



Leibniz International Proceedings in Informatics

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

## 1 Introduction

### 1.1 Hilbert Functions

Low-degree polynomials are fundamental objects in theoretical computer science, and their properties are extensively studied due to their important role in areas such as error correcting codes and circuit lower bounds. Let  $d \geq 0$  be an integer,  $\mathbb{F}$  be a field, and  $S \subseteq \mathbb{F}^n$  be a set. Each degree- $d$   $n$ -variate polynomial  $p$  over  $\mathbb{F}$  can be naturally viewed as a map  $p : \mathbb{F}^n \rightarrow \mathbb{F}$ , and hence also defines a map  $p|_S : S \rightarrow \mathbb{F}$ . Considering the linear space of all such maps in  $\mathbb{F}^S$ , which is a subspace of the space of all maps from  $S$  to  $\mathbb{F}$ , allows one to tap into a wide array of algebraic techniques in order to prove useful facts about the set  $S$ . This approach was for example utilized in complexity theory famously in the work of Smolensky [39], where proving bounds on the dimension of the aforementioned subspace was used to obtain lower-bounds for  $\text{AC}^0[\oplus]$  circuits computing the indicator function of the set  $S$ , for various  $S \subseteq \{0, 1\}^n$ . The dimension of the space consisting of  $p|_S$  for all degree- $d$  polynomials  $p$  is indeed a well-studied and classical concept in algebraic geometry known as the (affine) Hilbert function of  $S$ , denoted by  $h_S(d, \mathbb{F})$ . Hilbert functions encode important geometric and algebraic information, such as the dimension, degree, and regularity of varieties, in a more general context.

Hilbert functions have previously been studied in complexity theory due to their applications in circuit lower bounds, in particular for  $\text{AC}^0[\oplus]$  circuits, that were established by Smolensky [39] and Razborov [36]. Such applications require finding sets  $S \subseteq \{0, 1\}^n$  where the Hilbert function takes a very large value. However, it is also interesting to prove general lower bounds or find lower-bounding methods for arbitrary sets  $S$ . An example of such a result is the work of Moran and Rashtchian [32], who showed upper and lower bounds on  $h_S(d, \mathbb{F})$  for  $S \subseteq \{0, 1\}^n \subseteq \mathbb{F}^n$  via various concepts in VC theory. [32] treated the Hilbert function as a complexity measure of the set  $S$  and compared it to measures that arise naturally in learning theory, including “shattering” and “strong shattering” values.

Suppose  $r > 0$  is an integer. It is natural to wonder, what the extreme values of  $h_S(d, \mathbb{F})$  are, among all sets  $S$  of size  $|S| = r$ . It is not hard to show that the maximum value is equal to  $\min(r, h_{\mathbb{F}^n}(d, \mathbb{F}))$  when  $S \subseteq \mathbb{F}^n$ . For example, the maximum value of the Hilbert function of a set  $S \subseteq \mathbb{F}_2^n$  of size  $r$  is  $\min(r, \binom{n}{\leq d})$ .

On the other hand, finding the true smallest value of  $h_S(d, \mathbb{F})$  is a natural and intriguing question even when  $S$  is restricted to subsets of some finite and structured set in  $\mathbb{F}^n$ .

► **Question 1.** *Let  $0 \leq d \leq n$  be integers,  $\mathbb{F}$  be a field, and  $A = A_1 \times \dots \times A_n \subseteq \mathbb{F}^n$  where  $A_i \subseteq \mathbb{F}$  are finite sets. For any  $r \leq |A|$ , what is the smallest value of  $h_S(d, \mathbb{F})$  among all subsets  $S \subseteq A$  of cardinality  $|S| = r$ ?*

This question has been answered in the case of  $\mathbb{F} = \mathbb{F}_2$  and  $A = \mathbb{F}_2^n$  by Keevash and Sudakov [27] and Ben-Eliezer, Hod, and Lovett [6], and later generalized to  $\mathbb{F} = \mathbb{F}_p$  and  $A = \mathbb{F}_p^n$  by Beame, Oveis Gharan, and Yan [4]. For simplicity, let  $r = p^k$  for some  $k \geq 0$ . [4] proved that the smallest value of  $h_S(d, \mathbb{F}_p)$  with  $|S| = r$  is equal to the number of degree- $\leq d$  monomials on  $k$  variables, for example when  $p = 2$ , this is equal to simply  $\binom{k}{\leq d} = \binom{\log_2 r}{\leq d}$ .

We prove a more general result that answers Question 1 for arbitrary finite grids  $A \subseteq \mathbb{F}^n$  in arbitrary fields  $\mathbb{F}$ . We show that the smallest values of Hilbert functions are exactly determined by an extremal combinatorial question about the number of low-Hamming-weight elements in down-closed sets, which we solve by building on the work of Beelen and Datta [5].

The prior works discussed above were motivated by applications in bounding the list-size of the Reed-Muller codes and obtaining certain extensions of Frankl–Ray–Chaudhuri–Wilson theorems on cross-intersecting sets. In contrast, in this paper, we are interested in Question 1 due to its applications in pseudorandomness, particularly in randomness extraction.

Understanding the smallest values of Hilbert functions is closely related to the study of *degree- $d$  closure* of sets, a notion introduced by Nie and Wang [33].

► **Definition 1.** *The degree- $d$  closure of a set  $T \subseteq \mathbb{F}^n$  is defined as*

$$\text{cl}_d(T) := \{a \in \mathbb{F}^n \mid \text{for every degree-}d \text{ polynomial } f, f|_T \equiv 0 \Rightarrow f(a) = 0\}.$$

Equivalently,  $\text{cl}_d(T)$  is the set of all points  $a \in \mathbb{F}^n$  such that the values  $f(a)$  of a degree- $d$  polynomial  $f$  are determined by  $f|_T$ .

The existence of a small set with a large degree- $d$  closure has application to hitting-set generators for polynomials [17]. As an application of our answer to Question 1, we obtain an upper bound on the size of  $\text{cl}_d(T)$  in terms of  $|T|$ . Our bound in fact yields an optimal way of creating a small set with a large degree- $d$  closure.

Futhermore, Question 1 has direct implications to the theory of randomness extractors, which we discuss next.

## 1.2 Randomness Extractors

The theory of randomness extractors is an active research area that was initiated in [38, 7] with the motivation of simulating randomized algorithms with access to “weak” randomness sources. The main objective of this theory is to design *extractors* that are capable of purifying imperfect randomness sources into high-quality random bits or bit sequences. Extractors and related objects such as *dispersers*, *samplers*, and *condensers* have since found numerous applications in constructing other pseudorandom objects such as pseudorandom generators [34] and expander graphs [50], as well as applications in other areas of theoretical computer science and mathematics including cryptography [15], combinatorics [30], hardness of approximation [51], error correcting codes [42], and metric embeddings [25].

A deterministic extractor for a family  $\mathcal{X}$  of distributions over  $\{0, 1\}^n$  is a map  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  such that for any  $\mathbf{X} \in \mathcal{X}$ ,  $f(\mathbf{X})$  is close to the uniform distribution in statistical distance. It is common to measure the amount of randomness in a random variable  $\mathbf{X}$  by its min-entropy, defined  $H_\infty(\mathbf{X}) := -\log_2 \max_{x \in \{0, 1\}^n} \Pr[\mathbf{X} = x]$ . It is easy to show that no deterministic extractor can extract from general  $n$ -bit randomness sources of min-entropy as high as  $n - 1$  [11]. As a result, researchers in the area have explored two directions. Much of the focus in the area has been given to the more powerful *seeded extractors* that have access to an additional short purely random seed. This article contributes to another line of work that has extensively investigated the extra assumptions on the randomness sources that allow for explicit deterministic extractors and dispersers to exist. A widely studied class of sources in this latter direction, introduced in [43] is “samplable sources”, where the sources of randomness are distributions sampled by applying a low-complexity map (e.g., a decision forest, local map,  $\text{NC}^0$  circuit,  $\text{AC}^0$  circuit, an affine or a low-degree map) to the uniform distribution. Unfortunately, constructing explicit extractors even for sources samplable by really low-complexity maps has been quite challenging, and for example all the known constructions of extractors for local sources require quite high min-entropy of  $\Omega(\sqrt{n})$  [47, 14]. Due to the difficulty of constructing good explicit extractors and motivated by applications in complexity theory such as circuit lower bounds [29] and lower bounds for distribution-sampling [46], researchers have considered the seemingly easier task of proving the existence of low-complexity extractors [45, 19, 10, 44, 8, 16, 24, 12, 1].

The state of affairs is much worse when it comes to randomness extraction from sources sampled by more powerful maps such as  $\text{AC}^0[\oplus]$  or low-degree  $\mathbb{F}_2$ -polynomial maps. In this case obtaining nontrivial explicit constructions and even non-explicit low-complexity

extractors remains open. In fact, the same problems are open even in the case of dispersers. Here a map  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is a *disperser* for  $\mathcal{X}$  if for every source  $\mathbf{X} \in \mathcal{X}$ , the support of  $f(\mathbf{X})$  is  $\{0, 1\}$ . On the positive side, Chattopadhyay, Goodman, and Gurumukhani [9], recently proved the existence of deterministic (not necessarily low-complexity) extractors for low-degree  $\mathbb{F}_2$ -polynomial sources with logarithmic min-entropy.

### 1.3 Our Results on Hilbert Functions

We obtain our answer to Question 1 by first reducing it to a purely combinatorial problem. In particular, via an algebraic geometric argument, we prove the following theorem which states that the minimum value of Hilbert functions over subsets of a grid is exactly captured by a combinatorial quantity related to down-closed sets. (A set  $T \subseteq \mathbb{N}^n$  is said to be down-closed if  $T$  is closed under decreasing any coordinates of its elements.)

► **Theorem 2** (See Corollary 35). *Let  $\mathbb{F}$  be a field, and  $A_1, \dots, A_n \subseteq \mathbb{F}$  be finite sets of size  $|A_i| = r_i$ . Define  $A = A_1 \times \dots \times A_n$ . For every  $k \leq |A|$ ,*

$$\min_{S \subseteq A: |S|=k} h_S(d, \mathbb{F}) = \min_{\text{down-closed } T \subseteq F: |T|=k} |T_{\leq d}| ,$$

where  $F = \prod_i \{0, \dots, r_i - 1\}$  and  $T_{\leq d} = \{x \in T : \sum_i x_i \leq d\}$ .

For space reasons, we defer the proofs of Theorem 2 and the consequent results to the full version [20].

Let  $I$  be the ideal of  $\mathbb{F}[X_1, \dots, X_n]$  associated with a set  $S \subseteq A$ , that is, the set of all polynomials vanishing on  $S$ .

Classical results in algebraic geometry (such as Hilbert's Nullstellensatz) establish close connections between the structure of  $S$  and the structure of  $I$ , which allows us to focus on studying  $I$ .

The proof of Theorem 2 is based on the idea that the ideal  $I$  can be reasonably approximated by another ideal, *the ideal of leading terms of  $I$* . This approximation preserves important information about  $I$ , and consequently, about  $S$  as well. In particular, when the ideal of leading terms of  $I$  is defined with respect to a specific total order of monomials compatible with the total degree, it can be shown that such an approximation preserves the value of the Hilbert function. One advantage of working with the ideal of leading terms is that it is a *monomial ideal*, that is, an ideal generated by monomials, whose relatively simple structure can be analyzed using combinatorial tools.

We remark that the concept of transforming a general ideal into a monomial ideal is closely related to the theory of Gröbner bases, which serves as a basis of computational algebraic geometry. For a detailed discussion, see, e.g., [3]. This concept is also used in Smolensky's algebraic method for proving circuit lower bounds [40].

Theorem 2 allows us to reduce the problem of determining the smallest value of Hilbert function of a set of size  $k$  to understanding the smallest number of low-Hamming-weight points in down-closed sets of the same size. We then solve this combinatorial problem by proving that the minimum is obtained by the down-closed set  $M_F(k)$  which is defined as the set of  $k$  lexicographically first elements of  $F$ .

► **Theorem 3** (See Theorem 38). *Let  $1 \leq r_1 \leq \dots \leq r_n$  be integers and let  $F = \prod_{i=1}^n \{0, \dots, r_i - 1\}$ . Then*

$$\min_{\text{down-closed } T \subseteq F} |T_{\leq d}| = |M_F(k)_{\leq d}| .$$

In the case of  $r_1 = \dots = r_n = 2$ , we prove the above theorem via an elementary combinatorial argument, that via a series of operations turns any set of  $k$  elements into  $M_F(k)$  without increasing the number of elements of Hamming weight  $\leq d$ . We prove the general case by building on a recent result of Beelen and Datta [5]. This result generalizes the work of Wei [49] and Heijnen–Pelikaan [22, 21] in studying the generalized Hamming weights of certain linear codes.

We record the following corollary of our results specialized to finite fields which generalizes the bounds due to [27, 6, 4], where  $M_q^n(k)$  is the set of  $k$  lexicographically first strings in the set  $\prod_{i=1}^n \{0, 1, \dots, q-1\}$ .

► **Corollary 4** (See Corollary 39). *For every prime power  $q$ , and  $n, k, d \in \mathbb{N}$  where  $k \leq q^n$ , we have*

$$\min_{S \subseteq \mathbb{F}_q^n: |S|=k} h_S(d, \mathbb{F}_q) = |M_q^n(k)_{\leq d}|.$$

*In particular, setting  $q = 2$ , for every  $n, k, d \in \mathbb{N}$  where  $k \leq 2^n$ , and every  $S \subseteq \mathbb{F}_2^n$  of size  $|S| = k$ ,*

$$h_S(d, \mathbb{F}_2) \geq \binom{\lfloor \log(k) \rfloor}{\leq d}.$$

### 1.3.1 Degree- $d$ Closure of Sets

Motivated by its applications to combinatorial geometry, the notion of degree- $d$  closures of subsets of  $\mathbb{F}_q^n$  was introduced in [33]. This concept has since found further applications and connections to complexity theory [28, 35, 41] and pseudorandomness [17].

Recall that the degree- $d$  closure  $\text{cl}_d(T)$  of a set  $T \subseteq \mathbb{F}^n$  over a finite field  $\mathbb{F}$  is the set of all points  $a \in \mathbb{F}^n$  such that any degree- $d$  polynomial vanishing on  $T$  also vanishes at  $a$ . Nie and Wang [33] proved the following result.

► **Theorem 5** ([33, Theorem 5.6]). *Let  $n, d \in \mathbb{N}$  and  $T \subseteq \mathbb{F}_q^n$ . Then*

$$|\text{cl}_d(T)| \leq \frac{q^n}{h_{\mathbb{F}_q^n}(d, \mathbb{F}_q)} \cdot |T|.$$

Building on our results on Hilbert functions, we obtain an improvement of Theorem 5 by obtaining a tight upper bound on the size of degree- $d$  closures of sets.

► **Theorem 6** (See Theorem 47 and Theorem 48). *Let  $n, d, m \in \mathbb{N}$ . Let  $T \subseteq \mathbb{F}_q^n$  be a set of size  $m$ . Then*

$$|\text{cl}_d(T)| \leq \max_{0 \leq k \leq q^n: |M_q^n(k)_{\leq d}| \leq m} k = \begin{cases} \max_{0 \leq k \leq q^n: |M_q^n(k)_{\leq d}| = m} k & \text{if } m \leq h_{\mathbb{F}_q^n}(d, \mathbb{F}_q), \\ q^n & \text{otherwise.} \end{cases} \quad (1)$$

*Moreover, this bound is tight in the sense that for any  $0 \leq m \leq q^n$ , there exists  $T \subseteq \mathbb{F}_q^n$  of size  $m$  for which (1) holds with equality.*

In fact, the set  $T$  of size  $m$  that attains the bound in the above theorem can be constructed explicitly; see Theorem 48 for details.

For convenience, we state the following corollary which is used later in the paper. For  $n, d, \delta \in \mathbb{N}$ , denote by  $N(n, d, \delta)$  the number of monomials  $X_1^{e_1} \dots X_n^{e_n}$  with  $e_1, \dots, e_n \leq \delta$  and  $e_1 + \dots + e_n \leq d$ .

► **Corollary 7.** *Let  $n, d, \ell \in \mathbb{N}$ . If  $T \subseteq \mathbb{F}_q^n$  is a set of size less than  $N(\ell, d, q - 1)$ , then  $|\text{cl}_d(T)| < q^\ell$ . In particular, if  $q = 2$  and  $T \subseteq \mathbb{F}_2^n$  is a set of size less than  $\binom{\ell}{\leq d}$ , then  $|\text{cl}_d(T)| < 2^\ell$ .*

**Proof.** Observe that  $|\text{M}_q^n(q^\ell)_{\leq d}| = N(\ell, d, q - 1)$ . Then apply Theorem 6. ◀

Let us compare our bound with the bound of Nie and Wang in some specific settings.

► **Example 8.** Suppose  $\ell \leq n$ . Let  $T \subseteq \mathbb{F}_2^n$  be a set of size  $\binom{\ell}{\leq d} - 1$ . Then by Corollary 7, we have the bound  $|\text{cl}_d(T)| \leq 2^\ell - 1$ . On the other hand, the bound of Nie and Wang (Theorem 5) gives

$$|\text{cl}_d(T)| \leq \frac{2^n}{\binom{n}{\leq d}} \cdot |T|,$$

which is exponential in  $n$ , rather than in  $\ell$ , at least when  $d \leq (\frac{1}{2} - c)n$  for some constant  $c > 0$ .

► **Example 9.** Suppose  $\ell \leq n$  and  $d < q$ . Let  $T \subseteq \mathbb{F}_q^n$  be a set of size  $N(\ell, d, q - 1) - 1 = \binom{\ell+d}{d} - 1$ . Then by Corollary 7, we have the bound  $|\text{cl}_d(T)| \leq q^\ell - 1$ , which is exponential in  $\ell \log q$ . On the other hand, the bound of Nie and Wang (Theorem 5) gives

$$|\text{cl}_d(T)| \leq \frac{q^n}{\binom{n+d}{d}} \cdot |T|,$$

which is exponential in  $n \log q$ , rather than in  $\ell \log q$ , at least when  $n + d \leq q^{1-c}$  for some constant  $c > 0$ .

In [17], Doron, Ta-Shma, and Tell explicitly asked if there exists a small set  $T \subseteq \mathbb{F}_q^n$  whose degree- $d$  closure is very large. Our Theorem 6 gives an upper bound on the size of the degree- $d$  closure of  $T$  in terms of the size of  $T$ , which is tight in the sense that there exist sets  $T$  that exactly meet this bound for every cardinality of  $T$ . Moreover, such sets  $T$  can be constructed explicitly (see Theorem 48). Thus, we completely resolve the question posed by Doron, Ta-Shma, and Tell.

## 1.4 Our Results on Randomness Extractors

Continuing the line of work on low-complexity extractors, in this paper we investigate the power of low-degree polynomials in randomness extraction.

► **Question 2.** *For which families  $\mathcal{X}$  of sources does there exist a low-degree disperser? Similarly, for which families  $\mathcal{X}$  of sources does there exist a low-degree extractor?*

Let us first discuss the easier task of obtaining low-degree dispersers before moving on to our main application of low-degree extractors. For simplicity, we will focus on the most important case of extracting randomness over  $\mathbb{F}_2$ , but all our results easily generalize to  $\mathbb{F}_q$ . Non-explicit constructions of low-degree dispersers can be obtained via understanding the probability that a random low-degree polynomial is a disperser for a family  $\mathcal{X}$  of distributions over  $\{0, 1\}^n$  which we identify with  $\mathbb{F}_2^n$  in the natural way. Our starting point is the observation that the notion of Hilbert functions can be used to exactly describe the probability that a random degree- $d$  polynomial  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is a disperser for a fixed source  $\mathbf{X} \in \mathcal{X}$ . Indeed, this probability is exactly equal to  $1 - 2^{1-h_S(d, \mathbb{F}_2)}$ , where  $S = \text{support}(\mathbf{X})$ . Thus, in particular, Corollary 4 can be used to bound the probability that a random degree- $d$  polynomial is not a disperser for a fixed source.

### 1.4.1 Low-Degree Dispersers

Let  $\mathcal{X}$  be a family of sources of min-entropy  $\geq k$ . Observing that the support of any distribution  $\mathbf{X} \in \mathcal{X}$  is of size  $\geq 2^k$ , one gets as an immediate corollary of Corollary 4, the existence of low-degree dispersers  $\mathcal{X}$  as long as  $|\mathcal{X}|$  is small.

► **Theorem 10** (Informal, see Corollary 50). *Let  $n, d, k \geq 1$ . Let  $\mathcal{X}$  be a family of distributions of min-entropy  $\geq k$ . Then a random degree- $d$  polynomial over  $\mathbb{F}_2$  is a disperser for  $\mathcal{X}$  with probability at least*

$$1 - |\mathcal{X}| \cdot 2^{1 - \binom{k}{\leq d}}.$$

This theorem itself implies the existence of low-degree dispersers for several interesting families of samplable sources such as sources sampled by local maps, bounded-depth decision forests, and polynomial-sized bounded-fan-in circuits, to name a few.

A map  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$  is called  $\ell$ -local if each of its output bits depends on at most  $\ell$  input bits. A depth- $\ell$  decision forest is a map  $f$  where each output bit can be computed as a depth- $\ell$  decision tree. It is easy to obtain an upper bound exponential in  $\text{poly}(n)$  on the number of local or decision forest sources. Hence we get the following as a corollary of Theorem 10.

► **Corollary 11** (Informal, see Corollary 51). *Let  $1 \leq \ell \leq d \leq n$  be integers. There exists a degree- $d$  disperser*

- *for the family of  $\ell$ -local sources on  $\{0, 1\}^n$  with min-entropy  $k > d(2^\ell n + 2\ell n \log n)^{1/d}$ .*
- *for the family of depth- $\ell$  decision forest sources on  $\{0, 1\}^n$  with min-entropy  $k > d((\ell + \log n)2^{\ell+1}n)^{1/d}$ .*

As mentioned above, since in addition to the min-entropy requirement, the only requirement in Theorem 10 about the family  $\mathcal{X}$  is a bound on  $|\mathcal{X}|$ , it can be used to immediately obtain low-degree dispersers for various other families of sources as well. For example, since for any  $c$ , the number of Boolean circuits with  $\leq n^c$  bounded fan-in gates is at most  $2^{O(n^{c+1})}$ , one can also use Theorem 10 to obtain a degree- $O(c)$  disperser for such families of circuits. However, we will not do an exhaustive search for all such applications, and instead our main disperser applications will focus on two powerful families of sources, namely sources sampled by low-degree polynomials over  $\mathbb{F}_2$  and  $\text{AC}[\oplus]$  circuits which we define as the family of unbounded-depth polynomial-size Boolean circuits with AND, OR, XOR, NOT gates of unbounded fan-in, where the input gates are not counted towards the size.

Note that low-degree polynomial maps  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ , even affine ones, can depend on the entire input for any  $m \gg n$  and thus one cannot simply bound  $|\mathcal{X}|$  when  $\mathcal{X}$  is the family of sources sampled by low-degree polynomials. This property holds for  $\text{AC}[\oplus]$  circuits, as we allow them to non-trivially depend on an arbitrary number of input gates (since the circuit gates have unbounded fan-in). Nevertheless, utilizing an “input-reduction” trick of [9] which applies to both the foregoing families of sources, it can be shown that for our disperser purposes we may assume the input of both families of sources to be of length  $O(n)$ . This allows us to apply Theorem 10 to obtain low-degree dispersers for both of these families.

► **Theorem 12** (Informal, see Theorems 53 and 54). *For every  $1 \leq \ell < d \leq n$ , there exists a degree- $d$  disperser*

- *for the family of degree- $\ell$  sources on  $\{0, 1\}^n$  with min-entropy  $k \geq (12^\ell \cdot d^d \cdot n)^{\frac{1}{d-\ell}} + 1$ .*
- *for the family of  $n^\ell$ -size  $\text{AC}[\oplus]$  circuit sources on  $\{0, 1\}^n$  with min-entropy  $k \geq (30^2 \cdot d^d \cdot n^{2\ell})^{\frac{1}{d-2}} + 1$ .*

In particular, for every  $\ell \in \mathbb{N}$ , there is a degree- $(\ell + 2)$  disperser for degree- $\ell$  sources on  $\{0, 1\}^n$  with min-entropy  $\Omega(\sqrt{n})$ .

We note that both of these source families are very powerful, and to the best of our knowledge, no nontrivial low-complexity dispersers for either of these families of sources was known prior to this work (except in the easier case of degree-1 sources which corresponds to affine sources for which explicit extractors for logarithmic entropy was recently proved [30]). Let us also point out that the two foregoing classes have incomparable power, and that it is straightforward to use our proof technique to conclude the same result for a class of sources that generalizes both  $\text{AC}[\oplus]$  and constant-degree polynomials. Indeed, the input-reduction and counting idea works for the “hybrid” class of polynomial-size circuits which extends  $\text{AC}[\oplus]$  by allowing additional unbounded fan-in gates computing arbitrary polynomials of fixed constant degree. However, for ease of exposition, we have chosen to present only the results for  $\text{AC}[\oplus]$  and low-degree sources separately.

### 1.4.2 Low-Degree Extractors

Next, we move on to another application concerning the existence of low-degree extractors for samplable sources. Can we prove the existence of low-degree extractors for all the families for which we proved the existence of low-degree dispersers? We prove this by showing an analogue of Theorem 10 for extractors.

► **Theorem 13** (Informal, see Theorem 58 for the more general statement). *Let  $\mathcal{X}$  be a family of distributions of min-entropy  $k \geq 5 \log n$  over  $\{0, 1\}^n$  for large enough  $n$ . Then for every  $d \geq 6$ , a uniformly random degree- $d$  polynomial is an  $\varepsilon$ -extractor for  $\mathcal{X}$  with probability at least*

$$1 - |\mathcal{X}| \cdot e^{3n - O(k^{d/2})/n^2}$$

for  $\varepsilon = (2d)^d \cdot k^{-d/4}$ .

A similar statement (see Theorem 58) holds for families of sources that are close to convex combinations of another small family of sources. Combined with the input-reduction trick, we obtain as a corollary, the existence of low-degree extractors for various families of sources, notably, lower-degree sources and  $\text{AC}[\oplus]$  circuits.

► **Theorem 14** (Informal, see Theorem 60). *For all  $\ell, d \geq 1$ , and all large enough  $n$ , and  $k \geq 5 \log n$ . There exists a degree- $d$   $\mathbb{F}_2$ -polynomial that is an  $\varepsilon$ -extractor for the following families of sources over  $\{0, 1\}^n$  for  $\varepsilon = (2d)^d \cdot k^{-d/4}$ :*

- $\ell$ -local sources for  $k \geq (2^\ell n^3 \log n)^{2/d}$ .
- depth- $\ell$  decision forest sources for  $k \geq (2^\ell n^3 (\log n + \ell))^{2/d}$ .
- degree- $\ell$  sources for  $k \geq (3^\ell n)^{\frac{6}{d-2\ell}}$ .
- $n^\ell$ -size  $\text{AC}[\oplus]$  circuit sources (with unbounded number of input gates) for  $k \geq 3n^{\frac{4(\ell+1)}{d-4}}$ .

In Theorem 61, we further extend our low-degree extractors to multi-output extractors that output  $\Theta(k)$  bits. This is done by independently picking random degree- $d$  polynomials  $p_1, \dots, p_t$  for some  $t = \Theta(k)$ , and analyzing the probability that each  $p_i$  is an extractor for the family of sources obtained by  $\mathcal{X}$  conditioned on the values of  $p_1, \dots, p_{i-1}$ .

Let us now discuss our proof technique for Theorem 13. Recall that Theorem 10 was a corollary to Corollary 4 which showed that a random polynomial is with high probability non-constant on the support of any fixed high min-entropy distribution. A priori it is not clear how to use this bound on the Hilbert function to prove Theorem 13.

Indeed, let us consider the simpler case of a fixed  $k$ -flat source  $\mathbf{X}$  over  $\{0, 1\}^n$ , which is uniformly distributed over a set  $S \subseteq \{0, 1\}^n$  with  $|S| = 2^k$ . Note that a map  $p : \{0, 1\}^n \rightarrow \{0, 1\}$  is an extractor for  $\mathbf{X}$  if it has small bias on  $S$ . Thus, for example, to prove the special



case of Theorem 13 for small families of  $k$ -flat sources, we would need to prove that a random degree- $d$  polynomial is small-biased on  $S$  with high probability. However, Corollary 4 only tells us that  $h_S(d, \mathbb{F}_2) \geq \binom{k}{\leq d}$ , which is not enough to prove concentration bounds for the bias of a random degree- $d$  polynomial on an arbitrary set  $S$ . We note that when  $S$  is highly structured, that is when it is an affine subspace, this problem is equivalent to questions about list-decoding size of Reed-Muller codes, and known results such as one by Kaufman, Lovett, and Porat [26] that show that the number of distinct  $\varepsilon$ -biased degree- $d$  polynomials on a  $k$ -dimensional subspace  $S$  is at most  $(1/\varepsilon)^{k^{d-1}}$  could be utilized. However, for our application we have to deal with arbitrary sets  $S$ .

**Uniform covering by sets of full Hilbert dimension.** We say that a set  $T \subseteq \{0, 1\}^n$  has “full Hilbert dimension” if  $h_T(d, \mathbb{F}_2) = |T|$ . Note that when  $T$  has full Hilbert dimension, then the restriction of a random degree- $d$  polynomial to  $T$  is uniformly distributed over  $\{0, 1\}^T$ . In particular, if  $T$  is a sufficiently large set of full Hilbert dimension, then a random degree- $d$  polynomial is small-biased on  $T$  with high probability. We use this observation to design our technique for bounding the probability that a random degree- $d$  polynomial is small-biased on any fixed source  $\mathbf{X}$  of large min-entropy. For simplicity we describe the idea for flat sources. In this case,  $\mathbf{X}$  is uniformly distributed over a set  $S$  with  $|S| \geq 2^k$ . It is sufficient to prove the existence of an almost-uniform covering of  $S$  by large sets  $T_1, \dots, T_t$  of the same size with full Hilbert dimensions, where we call a covering almost-uniform if each element  $x \in S$  belongs to roughly  $tm/|S|$  many sets, where we assume  $|T_i| = m$ .

We obtain such a covering by analyzing the probability that a uniformly picked subset  $T_i \subseteq S$  has full Hilbert dimension. Using our bound on the Hilbert function, Corollary 4, which allows us to bound the size of the “degree- $d$  closure” of small sets, we prove that a random set  $T_i$  of size  $m$ , for some  $m = \binom{\Theta(k)}{\leq d}$ , has full Hilbert dimension with high probability. Similarly, we prove using the Bayes rule, that we may pick these good sets  $T_i$ ’s of full Hilbert dimension in a way that leads to an almost uniform covering. Since  $T_i$ ’s are of sufficiently large size  $\binom{\Theta(k)}{\leq d}$  and of full Hilbert dimension, we can use the Hoeffding inequality to bound the probability that a random degree- $d$  polynomial is biased on a  $T_i$  to be exponentially small in  $\Theta(k)^d$ , which is good enough for our applications to existence of low-degree extractors. We obtain the following result which can be used to prove Theorem 13.

► **Theorem 15** (Informal, see Theorem 57). *Let  $n, d, k \geq 1$ , and  $\varepsilon > 0$  be a real. Then for every distribution  $\mathbf{X}$  over  $\{0, 1\}^n$  with  $H_\infty(\mathbf{X}) \geq k$ , a uniformly random degree- $d$  polynomial  $f$  is an  $\varepsilon$ -extractor for  $\mathbf{X}$ , with probability at least  $1 - e^{3n - \varepsilon^2 \binom{\ell}{\leq d} / (Cn^2)}$  where  $\ell = k/2 - \log(32n/\varepsilon)$  and  $C = 7 \cdot (32)^2$ .*

We find our technique of obtaining almost uniform coverings with sets of full Hilbert dimension to be powerful, and hope that it will find other applications beyond the ones explored here.

## 1.5 Remarks

**Correlation bounds over arbitrary subsets.** We note that our proof of Theorem 15 (Theorem 58) can be modified to the following correlation bounds with any fixed function.

► **Theorem 16.** *Let  $n, d, k \geq 1$ ,  $\varepsilon > 0$  be a real, and  $g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  be a fixed function. Then for every distribution  $\mathbf{X}$  over  $\{0, 1\}^n$  with  $H_\infty(\mathbf{X}) \geq k$ , for a uniformly random degree- $d$  polynomial  $f$  we have*

$$\Pr[f(\mathbf{X}) = g(\mathbf{X})] = \frac{1}{2} \pm \varepsilon,$$

*with probability at least  $1 - e^{3n - \varepsilon^2 \binom{\ell}{\leq d} / (Cn^2)}$  where  $\ell = k/2 - \log(32n/\varepsilon)$  and  $C = 7 \cdot (32)^2$ .*

## 41:10 Hilbert Functions and Low-Degree Randomness Extractors

This generalization is quite straightforward, as once we obtain a uniform covering by sets of maximum Hilbert dimension, then Hoeffding bounds can be used to bound the correlation of a random polynomial with the fixed function restricted to the sets belonging to the cover. This can then be used to bound the over-all correlation with the fixed function in a similar way to the proof of Theorem 58.

**Punctured Reed-Muller codes.** The special case of Theorem 16 when  $\mathbf{X}$  is a flat source can be interpreted as a bound on the list-decoding size of Reed-Muller codes when “punctured” on a large set  $S \subseteq \mathbb{F}_2^n$ . Recall that the binary Reed-Muller code  $\text{RM}[d, n]$  consists of codewords in  $\mathbb{F}_2^n$  that correspond to the evaluation vectors of degree  $\leq d$  polynomials over  $\mathbb{F}_2$ . Given a set  $S \subseteq \mathbb{F}_2^n$ , the resulting punctured code consists of the evaluation of degree  $\leq d$  polynomials on  $S$ . In this context, Theorem 16 can be used to bound the list-size of any puncturing of the Reed-Muller code, showing that for any word  $w$  from  $\mathbb{F}_2^S$ , only a small fraction of codewords are within radius  $\frac{1}{2} - \varepsilon$  of  $w$ . Another interpretation of Theorem 16 is that any puncturing of the Reed-Muller codes over a set  $S$  can be turned into a “small-biased” code without much loss in the rate of the code.

**Sampling lower bound for polynomial sources.** Our low-degree extractor for lower-degree sources (Theorem 14) has a direct application in distributions that are hard to sample by low-degree polynomials. Indeed, an argument similar to the proof of [48, Lemma 3], Theorem 14 implies the existence of a degree- $O(d)$  polynomial  $p$  for which the distribution  $(\mathbf{U}, p(\mathbf{U}))$  cannot be sampled by any degree- $d$  source, where  $\mathbf{U} \sim \mathbf{U}_n$ .

Suppose that  $p$  is a degree- $O(d)$  polynomial that is an  $\varepsilon$ -extractor for the family of degree  $\leq 2d$  sources over  $\{0, 1\}^n$  of min-entropy  $\geq \frac{n}{2}$ , where  $\varepsilon = o(1)$ . The existence of such a polynomial  $p$  is guaranteed by Theorem 14. Now suppose that  $(G(\mathbf{U}'), g(\mathbf{U}'))$ , where  $\mathbf{U}' \sim \mathbf{U}_m$  for some  $m \geq 1$ , is a degree  $\leq d$  source sampling  $(\mathbf{U}, p(\mathbf{U}))$ . In particular,  $G$  is an  $n$ -bit degree  $\leq d$  source and  $g$  is a degree  $\leq d$  polynomial. Consider the  $n$ -bit random variable  $\mathbf{R} = G(\mathbf{U}') \cdot g(\mathbf{U}') + \mathbf{U}_n \cdot (1 - g(\mathbf{U}'))$ . Since  $\mathbf{R}$  is sampled by a degree  $\leq 2d$  source of min-entropy  $n - O(1)$ ,  $\Pr[p(\mathbf{R}) = 1] = \frac{1}{2} + o(1)$ . On the other hand, by the definition  $\mathbf{R}$ , we have  $\Pr[p(\mathbf{R}) = 1] \geq \frac{1}{2} + \Omega(1)$ , which is a contradiction.

**Related Work.** An independent and concurrent paper by Alrabiah, Goodman, Mosheiff, and Ribeiro [2] proves the existence of low-degree extractors for similar families of sources that are considered in our work, as well as sumset sources. While the proofs are quite different, they both rely on bounds on the dimension of punctured Reed-Muller codes (equivalently the Hilbert function).

## 2 Preliminaries

All logarithms in this paper are base 2. By  $\mathbb{N}$  we denote the set of non-negative integers. For a positive integer  $n$ , by  $[n]$  we denote the set  $\{1, \dots, n\}$ . For a prime power  $q$ , denote by  $\mathbb{F}_q$  the finite field  $q$  elements.

For simplicity, throughout this paper, we refer to a polynomial as a degree- $d$  polynomial if its total degree is *at most*  $d$ . When  $q$  is a prime power, by  $\mathcal{P}_q(n, d)$  we denote the set of all degree- $d$  polynomials from  $\mathbb{F}[X_1, \dots, X_n]$  with individual degrees at most  $q - 1$ . Note that each element of  $\mathcal{P}_q(n, d)$  corresponds to a unique map  $\mathbb{F}_q^n \rightarrow \mathbb{F}_q$ .

Let  $r_1, \dots, r_n \geq 1$  be integers and  $F = \prod_{i=1}^n \{0, \dots, r_i - 1\}$ . For  $x \in F$  and  $i \in [n]$ ,  $x_i$  denotes the  $i$ th coordinate of  $x$ . For  $x \in F$ , we define its *generalized Hamming weight* as  $|x| := \sum_i x_i$ , where the summation is over the integers. For an integer  $d \geq 0$ , and a set  $T \subseteq F$ , we denote the set of its elements of generalized Hamming weight  $\leq d$  by

$$T_{\leq d} := \{x \in T : |x| \leq d\}.$$

For  $a, b \in F$ , we write  $a \leq_P b$  if  $a_i \leq b_i$  for all  $i \in [n]$ . We say a subset  $T \subseteq F$  is *down-closed* if for all  $a, b \in F$  such that  $a \leq_P b$ , if  $b$  is in  $T$ , then so is  $a$ . Similarly, we say a subset  $T \subseteq F$  is *up-closed* if for all  $a, b \in F$  such that  $a \leq_P b$ , if  $a$  is in  $T$ , then so is  $b$ .

The *lexicographic order*  $\prec$  on  $F$  is defined as follows. For distinct  $x, y \in F$ ,  $x$  precedes  $y$ , denoted  $x \prec y$ , in lexicographic order if  $x_i < y_i$ , where  $i$  is the smallest index such that  $x_i \neq y_i$ .

We will be studying the following quantity.

► **Definition 17.** For  $F = \prod_{i=1}^n \{0, \dots, r_i - 1\}$  and  $k \leq |F|$ , let

$$\mathcal{H}_F(d, k) := \min_T |T_{\leq d}|,$$

where the minimum is taken over all down-closed sets  $T \subseteq F$  with  $|T| = k$ . Moreover, denote  $\mathcal{H}_F(d, k)$  by  $\mathcal{H}_q^n(d, k)$  in the special case where  $r_1 = \dots = r_n = q$  for some  $q \geq 1$ .

## 2.1 Probability Distributions

We use lowercase letters such as  $x, y$  to denote vectors, uppercase bold letters such as  $\mathbf{X}, \mathbf{Y}$  to denote random variables, and  $\mathcal{X}, \mathcal{Y}$  to denote families of distributions. By  $\mathbf{U}_n$  we denote the uniform distribution over  $\{0, 1\}^n$ .

The statistical distance between two distributions  $\mathbf{A}$  and  $\mathbf{B}$  over a finite domain  $X$  is

$$\Delta(\mathbf{A}, \mathbf{B}) = \frac{1}{2} \left( \sum_{x \in X} |\Pr[x \in \mathbf{A}] - \Pr[x \in \mathbf{B}]| \right).$$

We say two distributions  $\mathbf{A}$  and  $\mathbf{B}$  are  $\varepsilon$ -close if  $\Delta(\mathbf{A}, \mathbf{B}) \leq \varepsilon$ . For a distribution  $\mathbf{X} \sim \{0, 1\}^n$ , the min-entropy of  $\mathbf{X}$  is

$$H_\infty(\mathbf{X}) = \min_{x \in \text{support}(\mathbf{X})} -\log(\Pr[\mathbf{X} = x]).$$

We will use following forms of Chernoff's and Hoeffding's bounds (see, e.g., [31, 23]).

► **Theorem 18** (Chernoff bound). Let  $X_1, \dots, X_n \in \{0, 1\}$  be independent random variables. Let  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}(X)$ . Then we have

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}$$

for all  $0 < \delta < 1$ .

► **Theorem 19** (Hoeffding's inequality). Let  $X_1, \dots, X_n \in [0, 1]$  be independent random variables,  $X = \sum_{i=1}^n X_i$  and  $\mu = \mathbb{E}[X]$ . Then,

$$\Pr[|X - \mu| \geq R] \leq 2e^{-\frac{2R^2}{n}}.$$

## 2.2 Randomness Sources, Dispersers, and Extractors

► **Definition 20** (Sources and Their Convex Combinations). *A distribution  $\mathbf{X} \sim \{0, 1\}^n$  is a source from a class  $\mathcal{C}$  of functions, if  $\mathbf{X} = f(\mathbf{U}_m)$  for some  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n \in \mathcal{C}$ . A distribution  $\mathbf{Y}$  is a convex combination of sources  $\mathbf{X}_i$  if  $\mathbf{Y} = \sum_i p_i \mathbf{X}_i$  for some non-negative  $p_i$  satisfying  $\sum_i p_i = 1$ , i.e.,  $\mathbf{Y}$  samples from each  $\mathbf{X}_i$  with probability  $p_i$ .*

One of the most powerful classes of sources that we consider in this work is the class of circuits of polynomial size.

► **Definition 21** ( $\text{AC}[\oplus]$  circuits). *An  $\text{AC}[\oplus]$  circuit is an unbounded-depth Boolean circuit consisting of AND, OR, XOR, NOT gates of unbounded fan-in. The size of such a circuit is the number of non-input gates in it.*

We focus on the class of  $\text{AC}[\oplus]$  circuit as it generalizes circuit classes previously studied in this context: unbounded-depth circuits of bounded fan-in from  $\text{P/poly}$ , and bounded-depth circuits of unbounded fan-in from, say,  $\text{AC}^0$ . We remark that we define  $\text{AC}[\oplus]$  sources (see Definition 22) as sources where each output is computed by an  $\text{AC}[\oplus]$  circuit of polynomial size but with an arbitrary (possibly super-polynomial) number of inputs. This explains why in this context  $\text{P/poly}$  and  $\text{AC}^0$  circuits are incomparable, and why we work with  $\text{AC}[\oplus]$  circuits generalizing both of the aforementioned classes. In fact, our results hold even for a larger class of circuits where not only XOR but arbitrary constant-degree polynomials over  $\mathbb{F}_2$  can be computed at gates (see the discussion at the end of Section 6).

► **Definition 22** (Structured Sources). *Let  $n, d, m \in \mathbb{N}$ ,  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ , and  $\mathbf{X}$  be a distribution over  $\{0, 1\}^n$  that is generated as  $f(\mathbf{U}_m)$ .*

- $\mathbf{X}$  is called a  $d$ -local source if every output bit of  $f$  depends only on at most  $d$  of its input bits.
- $\mathbf{X}$  is called a depth- $d$  decision forest source if every output bit of  $f$  is determined by a depth- $d$  decision tree of its input variables.
- $\mathbf{X}$  is called a degree- $d$  source if every output bit of  $f$  is a degree- $d$  polynomial over  $\mathbb{F}_2$ .
- $\mathbf{X}$  is called a size- $n^d$  circuit source if there is an  $\text{AC}[\oplus]$  circuit of size  $n^d$  that computes all output bits of  $f$ .

Note that every  $d$ -local source is a depth- $d$  decision forest source, and a degree- $d$  source. Also, every depth- $d$  decision forest source is a degree- $d$  source and a  $2^d$ -local source.

We will use the following bounds on the numbers of  $d$ -local sources and depth- $d$  decision forest sources.

► **Proposition 23.** *Let  $n, d \geq 1$ .*

- *The number of  $d$ -local sources over  $\{0, 1\}^n$  is bounded from above by  $2^{2^d n + 2dn \log n}$ .*
- *The number of depth- $d$  decision forest sources is bounded from above by  $2^{(d+\log n)2^{d+1}n}$ .*

For polynomial and circuit sources where the number of input bits cannot be bounded by a small function of  $n$  (unlike the sources considered in Proposition 23), we will need the following bounds on the number of such sources for a fixed number of input bits  $m$ .

► **Proposition 24.** *Let  $n, d, m \geq 1$ .*

- *The number of degree- $d$  polynomials  $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$  is bounded from above by  $2^{n \binom{m}{\leq d}}$ .*
- *The number of  $\text{AC}[\oplus]$  circuits  $C : \{0, 1\}^m \rightarrow \{0, 1\}^n$  of size  $n^d$  is bounded from above by  $2^{4n^d(n^d+m)}$ .*

► **Definition 25** (Disperser). *A function  $\text{Disp} : \{0, 1\}^n \rightarrow \{0, 1\}$  is a disperser for a family  $\mathcal{X}$  of sources over  $\{0, 1\}^n$  with min-entropy  $k$ , if for every source  $\mathbf{X} \in \mathcal{X}$  with  $H_\infty(\mathbf{X}) \geq k$ , the support of  $\text{Disp}(\mathbf{X})$  is  $\{0, 1\}$ .*

► **Definition 26** (Extractor). *A function  $\text{Ext} : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is an  $\varepsilon$ -extractor for a family  $\mathcal{X}$  of sources over  $\{0, 1\}^n$  with min-entropy  $k$ , if for every source  $\mathbf{X} \in \mathcal{X}$  with  $H_\infty(\mathbf{X}) \geq k$ ,  $\Delta(\text{Ext}(\mathbf{X}(U_t)), U_m) \leq \varepsilon$ .*

For clarity of presentation, in this paper when working with sources that are guaranteed to have entropy  $H_\infty(\mathbf{X}) \geq k$ , we will always assume that  $k$  is an integer.

### 2.3 Hilbert Functions and Standard Monomials

In this section, we recall some necessary definitions (see, e.g., [13]). Let  $\mathbb{F}$  be a field,  $X_1, \dots, X_n$  be indeterminates, and  $\mathbb{F}[X_1, \dots, X_n]$  be the polynomial ring in  $n$  indeterminates over  $\mathbb{F}$ . For a polynomial  $f \in \mathbb{F}[X_1, \dots, X_n]$  and  $S \subseteq \mathbb{F}^n$ , let  $f|_S \in \mathbb{F}^S$  be the restriction of  $f$  to  $S$ . For  $d \in \mathbb{N}$ , by  $\Gamma_S(d) \subseteq \mathbb{F}^S$  we denote the vector space spanned by  $f|_S$  for all degree- $d$  polynomials  $f$ :

$$\Gamma_S(d) := \{f|_S : f \in \mathbb{F}[X_1, \dots, X_n], \deg(f) \leq d\}.$$

► **Definition 27** (Hilbert function). *For a set  $S \subseteq \mathbb{F}^n$ , the (affine) Hilbert function of  $S$  over  $\mathbb{F}$ ,  $h_S(\cdot, \mathbb{F}) : \mathbb{N} \rightarrow \mathbb{N}$ , is defined as the dimension of  $\Gamma_S(d)$  over  $\mathbb{F}$ , i.e.,*

$$h_S(d, \mathbb{F}) := \dim_{\mathbb{F}}(\Gamma_S(d)).$$

► **Definition 28** (Monomial order). *Let  $\preceq$  be a total order on the monomials in a polynomial ring  $\mathbb{F}[X_1, \dots, X_n]$ . The order  $\preceq$  is called a monomial order if 1 is the minimal element of  $\preceq$ , and for all monomials  $m_1, m_2, m$  satisfying  $m_1 \preceq m_2$ , we have that  $m_1 m \preceq m_2 m$ . The order  $\preceq$  is degree-compatible if for all monomials  $m_1, m_2$  such that  $\deg(m_1) < \deg(m_2)$ , we have that  $m_1 \preceq m_2$ .*

Examples of degree-compatible monomial orders include the graded lexicographic and graded reverse lexicographic orders.

► **Definition 29** (Graded orders). *The graded lexicographic order  $\leq_{\text{grlex}}$  and the graded reverse lexicographic order  $\leq_{\text{grevlex}}$  are defined as follows. For a pair of monomials  $m_1 = X_1^{\alpha_1} \dots X_n^{\alpha_n}$  and  $m_2 = X_1^{\beta_1} \dots X_n^{\beta_n}$ , let  $\alpha = \sum_{i=1}^n \alpha_i$ ,  $\beta = \sum_{i=1}^n \beta_i$ , and  $\gamma = (\beta_1 - \alpha_1, \dots, \beta_n - \alpha_n)$ . We have that  $m_1 \leq_{\text{grlex}} m_2$  if and only if either  $\alpha < \beta$ , or  $\alpha = \beta$  and the leftmost non-zero entry of  $\gamma$  is positive. Similarly,  $m_1 \leq_{\text{grevlex}} m_2$  if and only if either  $\alpha < \beta$ , or  $\alpha = \beta$  and the rightmost non-zero entry of  $\gamma$  is negative.*

► **Definition 30** (Leading monomial). *For a nonzero polynomial  $f \in \mathbb{F}[X_1, \dots, X_n]$ , the leading monomial of  $f$  under a monomial order  $\preceq$  is the largest monomial of  $f$  under  $\preceq$ .*

Let  $R$  be a commutative ring (such as the polynomial ring  $\mathbb{F}[X_1, \dots, X_n]$ ). An ideal of  $R$  is a subset  $I$  of  $R$  such that for all  $a, b \in I$  and  $r \in R$ , we have that  $a + b \in I$  and  $ra \in I$ .

► **Definition 31** (Standard monomial). *Let  $I$  be an ideal of  $\mathbb{F}[X_1, \dots, X_n]$ , and  $\preceq$  be a monomial order. A standard monomial  $m$  of  $I$  is a monomial in  $X_1, \dots, X_n$  that is not the leading monomial of any nonzero polynomial in  $I$ .*

## 41:14 Hilbert Functions and Low-Degree Randomness Extractors

For an ideal  $I$  and  $d \in \mathbb{N}$ ,  $\text{SM}(I)$  denotes the set of all standard monomials of  $I$ , and  $\text{SM}_{\leq d}(I)$  denotes the set of all standard monomials of  $I$  of degree at most  $d$ :

$$\text{SM}_{\leq d}(I) = \{m \in \text{SM}(I) : \deg(m) \leq d\}.$$

For a set  $S \subseteq \mathbb{F}^n$ , by  $I(S)$  we denote the ideal of polynomials in  $\mathbb{F}[X_1, \dots, X_n]$  vanishing on  $S$ ,

$$I(S) = \{f \in \mathbb{F}[X_1, \dots, X_n] : f|_S = 0^S\}.$$

For an ideal  $I$  of  $\mathbb{F}[X_1, \dots, X_n]$ , define the set  $V(I) \subseteq \mathbb{F}^n$  by

$$V(I) = \{a \in \mathbb{F}^n : f(a) = 0 \text{ for all } f \in I\}.$$

By definition, for all  $f \in I$  and  $a \in V(I)$ , we have  $f(a) = 0$ . So  $I \subseteq I(V(I))$ .

Finally, for a set  $S \subseteq \mathbb{F}^n$ , define

$$\text{SM}(S) = \text{SM}(I(S)) \quad \text{and} \quad \text{SM}_{\leq d}(S) = \text{SM}_{\leq d}(I(S)).$$

We say that a set  $T$  of monomials is *down-closed* if for all monomials  $m$  and  $m'$  such that  $m \in T$  and  $m'$  divides  $m$ , it holds that  $m' \in T$ . It is easy to see that  $\text{SM}(S)$  is down-closed. Indeed, if  $m'$  was the leading monomial of a polynomial  $p \in I(S)$ , then  $m$  would be the leading monomial of the polynomial  $p \cdot (m/m') \in I(S)$ .

We will use the following facts about  $\text{SM}(S)$  and  $\text{SM}_{\leq d}(S)$ , which are proven, for example, in [37, Lemma 1] and [18, Corollary 2.1.21].

► **Lemma 32.** *Let  $S \subseteq \mathbb{F}^n$  be a finite set. Then*

(a) *for every monomial order  $\preceq$ ,*

$$|S| = |\text{SM}(S)|;$$

(b) *for every degree-compatible monomial order  $\preceq$  and every  $d \in \mathbb{N}$ ,*

$$h_S(d, \mathbb{F}) = |\text{SM}_{\leq d}(S)|.$$

### 3 Hilbert Functions of Sets in Finite Grids

Let  $\mathbb{F}$  be a field. We consider Hilbert functions of subsets of a finite grid  $A = \prod_{i=1}^n A_i$ , where each  $A_i$  is a finite subset of the field  $\mathbb{F}$ . The main result of this section is that the minimum value  $h_S(d, \mathbb{F})$  of a set  $S \subseteq A$  of size  $k$  equals the quantity  $\mathcal{H}_F(d, k)$  introduced in Definition 17, where  $F = \prod_{i=1}^n \{0, 1, \dots, |A_i| - 1\}$ .

Consider the following setting: Let  $r_1, \dots, r_n$  be integers such that  $1 \leq r_i \leq |\mathbb{F}|$  for  $i \in [n]$ . For each  $i \in [n]$ , let  $A_i$  be a subset of  $\mathbb{F}$  consisting of  $r_i$  distinct elements  $a_{i,1}, \dots, a_{i,r_i} \in \mathbb{F}$ . Let  $A$  be the Cartesian product  $\prod_{i=1}^n A_i$ . Let  $\mathcal{M}$  be the set of monomials dividing  $\prod_{i=1}^n X_i^{r_i-1}$ . Let  $\sigma_A$  be the bijection from  $\mathcal{M}$  to  $A$  defined by

$$\sigma_A : \prod_{i=1}^n X_i^{e_i} \mapsto (a_{1,e_1+1}, \dots, a_{n,e_n+1}). \quad (2)$$

Finally, fix a degree-compatible monomial order  $\preceq$ .

The next lemma states that every down-closed subset  $T \subseteq \mathcal{M}$  can be realized as the set of standard monomials of the set  $\sigma_A(T) \subseteq A$ .

► **Lemma 33.** *Let  $T$  be a down-closed subset of  $\mathcal{M}$ .*

*Then  $\text{SM}(\sigma_A(T)) = T$ .*

For space reasons, we defer the proofs of Lemma 33 and the consequent results to the full version [20].

► **Lemma 34.** *Let  $k, d \in \mathbb{N}$  such that  $k \leq |A|$ . Then*

$$\min_{S \subseteq A: |S|=k} h_S(d, \mathbb{F}) = \min_{\text{down-closed } T \subseteq \mathcal{M}: |T|=k} |\{m \in T : \deg(m) \leq d\}|.$$

Let  $F = \prod_{i=1}^n \{0, 1, \dots, r_i - 1\}$ . Let  $\phi : \mathcal{M} \rightarrow F$  be the bijection

$$\phi : \prod_{i=1}^n X_i^{e_i} \mapsto (e_1, \dots, e_n). \quad (3)$$

Lemma 34 can now be reformulated as follows.

► **Corollary 35.** *Let  $k, d \in \mathbb{N}$  such that  $k \leq |A|$ . Then*

$$\min_{S \subseteq A: |S|=k} h_S(d, \mathbb{F}) = \mathcal{H}_F(d, k). \quad (4)$$

For the special case of a finite field  $\mathbb{F} = \mathbb{F}_q$ ,  $r_1 = \dots = r_n = q$ , and  $A_1 = \dots = A_n = \mathbb{F}_q$ , we have  $A = \mathbb{F}_q^n$ , and the right-hand side of Equation (4) becomes  $\mathcal{H}_q^n(d, k)$  from Definition 17. This leads us to the following corollary.

► **Corollary 36.** *For every  $n, k, d \in \mathbb{N}$  where  $k \leq q^n$ , a prime power  $q$ , and every set  $S \subseteq \mathbb{F}_q^n$  of size  $|S| = k$ , we have that*

$$h_S(d, \mathbb{F}_q) \geq \mathcal{H}_q^n(d, k).$$

Finally, we state the following lemma, which will be used in Section 5. Its proof reuses ideas from the previous proofs in this section.

► **Lemma 37.** *Let  $n, d \in \mathbb{N}$ . Let  $\sigma_A : \mathcal{M} \rightarrow A$  and  $\phi : \mathcal{M} \rightarrow F$  be the bijections (2) and (3) respectively. Let  $S \subseteq A$  such that  $T := \sigma_A^{-1}(S) \subseteq \mathcal{M}$  is down-closed. Let  $T' = \phi(T) \subseteq F$ . Then  $h_S(d, \mathbb{F}) = T'_{\leq d}$ .*

## 4 Number of Points with Low Hamming Weight in Down-Closed Sets

In this section, we will find the exact values of all  $\mathcal{H}_q^n(d, k)$  which, by Corollary 36, will give us tight lower bounds on the Hilbert function of sets of size  $k$ .

For every  $n, k, q$  where  $k \leq q^n$ , we define  $M_q^n(k)$  as the set of the first  $k$  elements of  $\{0, \dots, q-1\}^n$  in lexicographic order.

The main result of this section is the following theorem.

► **Theorem 38.** *For every  $n, k, d, q \in \mathbb{N}$  where  $k \leq q^n$ ,*

$$\mathcal{H}_q^n(d, k) = |M_q^n(k)_{\leq d}|.$$

Combining Corollary 36 and Theorem 38, we obtain the following bounds on the Hilbert function.

## 41:16 Hilbert Functions and Low-Degree Randomness Extractors

► **Corollary 39.** For every prime power  $q$ , and  $n, k, d \in \mathbb{N}$  where  $k \leq q^n$ , we have

$$\min_{S \subseteq \mathbb{F}_q^n: |S|=k} h_S(d, \mathbb{F}_q) = |M_q^n(k)_{\leq d}|.$$

In particular, setting  $q = 2$ , for every  $n, k, d \in \mathbb{N}$  where  $k \leq 2^n$ , and every  $S \subseteq \mathbb{F}_2^n$  of size  $|S| = k$ ,

$$h_S(d, \mathbb{F}_2) \geq \binom{\lfloor \log(k) \rfloor}{\leq d}.$$

We will use the following notation: For  $t \in \{0, 1, \dots, n\}$ , define  $\mathcal{D}_q^n(t)$  to be the set of  $x \in \{0, \dots, q-1\}^n$  whose first  $n-t$  coordinates are zero.

Note that for every  $q, k$ , and  $n$ ,

$$\mathcal{D}_q^n(\lfloor \log_q k \rfloor) \subseteq M_q^n(k) \subseteq \mathcal{D}_q^n(\lceil \log_q k \rceil).$$

When  $n$  and  $q$  are clear from the context, we omit the superscript  $n$  and the subscript  $q$  from  $M_q^n(k)$ ,  $\mathcal{D}_q^n(t)$ , and  $\mathcal{H}_q^n(d, k)$ .

### 4.1 The Boolean Case, $q = 2$

For a set  $S \subseteq \{0, 1\}^n$ , let  $\min(S)$  and  $\max(S)$  be respectively the smallest and the largest strings in  $S$  in lexicographic order. We say a set  $S \subseteq \{0, 1\}^n$  is a *contiguous  $k$ -set* if  $|S| = k$  and  $S$  consists of all  $x$  such that  $\min(S) \preceq x \preceq \max(S)$ .

We first show that  $M(k)$  has the largest number of low Hamming weight strings among all contiguous  $k$ -sets.

► **Lemma 40.** Let  $n, k, d \in \mathbb{N}$  be integers such that  $k \leq 2^n$ . Let  $S^k \subseteq \{0, 1\}^n$  be a contiguous  $k$ -set. Then  $|M(k)_{\leq d}| \geq |S_{\leq d}^k|$ .

We now use Lemma 40 to prove that if a contiguous  $k$ -set  $S^k$  that does *not* contain any of the first  $k$  strings in lexicographic order, then the result of Lemma 40  $|S_{\leq d}^k| \leq |M(k)_{\leq d}|$  can be strengthened to  $|S_{\leq d}^k| \leq |M(k)_{\leq d-1}|$ .

► **Lemma 41.** Let  $n, k, d \in \mathbb{N}$  be integers such that  $k \leq 2^n$ . Let  $S^k \subseteq \{0, 1\}^n$  be a contiguous  $k$ -set. If  $S^k \cap M(k) = \emptyset$ , then  $|M(k)_{\leq d-1}| \geq |S_{\leq d}^k|$ .

We are finally ready to prove the Boolean case of Theorem 38.

**Proof of Theorem 38, the  $q = 2$  case.** Let  $S \subseteq \{0, 1\}^n$  be a down-closed set of size  $k$ . We prove this theorem by a simultaneous induction on  $k, d \geq 0$ .

For the base cases, we consider pairs  $(k, d)$  such that  $d = 0$  or  $k \leq 2^d$ . The case of  $d = 0$  is trivial. For the case where  $k \leq 2^d$ , a down-closed set  $S$  of size  $k$  cannot have strings of Hamming weight  $> d$ , thereby showing  $|S_{\leq d}| = k$ . Also, by construction,  $M(k)$  is a down-closed set of size  $k$ , implying  $\mathcal{H}(d, k) = |M(k)_{\leq d}| = k$  in this case.

Given  $d \geq 1$  and  $k > 2^d$ , assume that the theorem is true for all  $(k', d')$  such that either  $k' < k$ , or  $k' = k$  and  $d' < d$ . Suppose  $S$  is a down-closed set of size  $k$  and let  $m$  be the smallest integer such that  $S \subseteq \mathcal{D}(m)$ . Define

$$\begin{aligned} S^0 &:= \{x \in S : x_{n-m+1} = 0\}, \\ S^1 &:= \{x - e_{n-m+1} : x \in S \text{ and } x_{n-m+1} = 1\}. \end{aligned}$$



Since  $S$  is down-closed, we have  $S^1 \subseteq S^0$ . Moreover,

$$|S_{\leq d}| = |S_{\leq d}^0| + |S_{\leq d-1}^1|.$$

Applying the induction hypothesis for  $k' = |S^0| < k$  and  $d$ , we get  $|\mathsf{M}(|S^0|)_{\leq d}| \leq |S_{\leq d}^0|$ . Let  $T = \mathsf{M}(k) \setminus \mathsf{M}(|S^0|)$ . Since  $|S^1| \leq |S^0|$ , we have  $\mathsf{M}(|S^1|) \cap T = \emptyset$ , and we may apply Lemma 41 to get  $|T_{\leq d}| \leq |\mathsf{M}(|S^1|)_{\leq d-1}|$ . Now applying the induction hypothesis for  $k' = |S^1|$  and  $d' = d - 1$ , we get  $|\mathsf{M}(|S^1|)_{\leq d-1}| \leq |S_{\leq d-1}^1|$ . Combining these observations, we get

$$\begin{aligned} |\mathsf{M}(k)_{\leq d}| &= |\mathsf{M}(|S^0|)_{\leq d}| + |T_{\leq d}| \\ &\leq |S_{\leq d}^0| + |\mathsf{M}(|S^1|)_{\leq d-1}| \\ &\leq |S_{\leq d}^0| + |S_{\leq d-1}^1| \\ &= |S_{\leq d}|. \end{aligned}$$

This concludes the induction, and shows that for every  $k, d \geq 0$ , and down-closed set  $S$  of size  $k$ ,  $|\mathsf{M}(k)_{\leq d}| \leq |S_{\leq d}|$ .  $\blacktriangleleft$

## 4.2 The General Case of Finite Grids

We prove Theorem 38 in this subsection. In fact, we prove the theorem in a more general setting, described as follows.

Let  $F = \prod_{i=1}^n \{0, 1, \dots, r_i - 1\}$  where  $r_1 \leq r_2 \leq \dots \leq r_n$ . Let  $d \in \mathbb{N}$ . We introduce the following notations:

For  $S \subseteq F$ , define  $\nabla(S) := \{a \in F : b \leq_P a \text{ for some } b \in S\}$ , i.e.,  $\nabla(S)$  is the up-closure of  $S$ . For  $k \in \{0, \dots, |F|\}$ , denote by  $\mathsf{M}(k)$  the set of the smallest  $k$  elements of  $F$  in lexicographic order. And for  $r \in \{0, \dots, |F_{\leq d}|\}$ , denote by  $\mathsf{L}_{\leq d}(r)$  the set of the largest  $r$  elements of  $F_{\leq d}$  in lexicographic order.

The main result of this subsection is the following generalization of Theorem 38.

► **Theorem 42.** *For every  $k \in \mathbb{N}$  such that  $k \leq |F|$ ,*

$$\mathcal{H}_F(d, k) = |\mathsf{M}(k)_{\leq d}|.$$

We derive Theorem 42 from a combinatorial result of Beelen and Datta [5], which generalizes the earlier work of Wei [49] and Heijnen–Pellikaan [22, 21].

► **Theorem 43** ([5, Theorem 3.8]). *Let  $S \subseteq F_{\leq d}$  and  $r = |S|$ . Then  $|\nabla(\mathsf{L}_{\leq d}(r))| \leq |\nabla(S)|$ .<sup>1</sup>*

Define  $\Delta(S) := F \setminus \nabla(S)$  for  $S \subseteq F$ . The next lemma gives a characterization of  $\Delta(S)$ .

► **Lemma 44.** *Let  $T \subseteq F_{\leq d}$  be down-closed and  $S = F_{\leq d} \setminus T$ . Then  $\Delta(S)$  is the unique maximal set with respect to inclusion among all down-closed subsets  $U$  of  $F$  satisfying  $U_{\leq d} = T$ .*

► **Lemma 45.** *Let  $r \in \{0, \dots, |F_{\leq d}|\}$  and  $k = |\Delta(\mathsf{L}_{\leq d}(r))|$ . Then  $\Delta(\mathsf{L}_{\leq d}(r)) = \mathsf{M}(k)$ .*

<sup>1</sup> In [5],  $\mathsf{L}_{\leq d}(r)$  is denoted by  $M(r)$ , while we use  $\mathsf{M}(r)$  to denote the set of the smallest  $r$  elements of  $F$  in lexicographic order.

## 5 A Tight Bound on the Size of Degree- $d$ Closures of Sets

For  $n, d, \delta \in \mathbb{N}$ , denote by  $N(n, d, \delta)$  the number of monomials  $X_1^{e_1} \cdots X_n^{e_n}$  with  $e_1, \dots, e_n \leq \delta$  and  $e_1 + \cdots + e_n \leq d$ . For example,  $N(n, d, 1) = \binom{n}{\leq d}$  and  $N(n, d, \delta) = \binom{n+d}{d}$  for  $d \leq \delta$ .

► **Lemma 46.**  $h_{\mathbb{F}_q^n}(d, \mathbb{F}_q) = N(n, d, q-1)$ .

In particular, Theorem 5, which was proved by Nie and Wang [33], can be restated as

$$|\text{cl}_d(T)| \leq \frac{q^n}{h_{\mathbb{F}_q^n}(d, \mathbb{F}_q)} \cdot |T| = \frac{q^n}{N(n, d, q-1)} \cdot |T|. \quad (5)$$

We now give the following tight bound on the size of the degree- $d$  closure of a set  $T \subseteq \mathbb{F}_q^n$ , improving (5).

► **Theorem 47.** *Let  $n, d, m \in \mathbb{N}$ . Let  $T \subseteq \mathbb{F}_q^n$  be a set of size  $m$ . Then*

$$|\text{cl}_d(T)| \leq \max_{0 \leq k \leq q^n: |\mathbb{M}_q^n(k)_{\leq d}| \leq m} k = \begin{cases} \max_{0 \leq k \leq q^n: |\mathbb{M}_q^n(k)_{\leq d}| = m} k & \text{if } m \leq N(n, d, q-1), \\ q^n & \text{otherwise.} \end{cases} \quad (6)$$

The next theorem states that the bound in Theorem 47 is tight and explicitly constructs sets that meet this bound.

► **Theorem 48.** *Let  $\sigma_A : \mathcal{M} \rightarrow A$  and  $\phi : \mathcal{M} \rightarrow F$  be the bijections (2) and (3) respectively, where  $A = \mathbb{F}_q^n$ ,  $F = \{0, 1, \dots, q-1\}^n$ , and  $\mathcal{M} = \{\prod_{i=1}^n X_i^{e_i} : 0 \leq e_1, \dots, e_n \leq q-1\}$ . Let  $m$  be any integer such that  $0 \leq m \leq q^n$ . Choose the maximum  $k \leq q^n$  such that  $|\mathbb{M}_q^n(k)_{\leq d}| \leq m$ . Let  $T_0 = (\sigma_A \circ \phi^{-1})(\mathbb{M}_q^n(k)_{\leq d}) \subseteq A = \mathbb{F}_q^n$ . If  $|T_0| \geq m$ , let  $T = T_0$ . Otherwise, let  $T$  be an arbitrary set obtained by adding  $m - |T_0|$  elements from  $\mathbb{F}_q^n \setminus T_0$  to  $T_0$ . Then  $T$  is a set of size  $m$  that attains the equality in (6).*

## 6 Low-Degree Dispersers

In this section, we will show how to use Theorem 38 to conclude the existence of low-degree dispersers for various families of sources. In Section 6.1, we will use Corollary 39 to show that for every family of at most  $2^{O(k^d)}$  sources of min-entropy  $k$ , there exists a degree- $d$  disperser. In particular, this will imply dispersers for local sources and bounded-depth decision forest sources. In Section 6.2, we will extend this result to large families of sources, including polynomial and circuit sources.

### 6.1 Dispersers for Small Families of Sources

In Theorem 49, we use the bound of Corollary 39 on the values of Hilbert functions to bound the probability that a random polynomial takes a fixed value on an arbitrary subset of  $\mathbb{F}_2^n$ .

► **Theorem 49.** *Let  $n, d \geq 1$ ,  $S \subseteq \mathbb{F}_2^n$  be an arbitrary nonempty set, and  $f : S \rightarrow \mathbb{F}_2$  be a function. Then,*

$$\Pr_{p \in_u \mathcal{P}_2(n, d)} [p|_S \equiv f] \leq 2^{-h_S(d, \mathbb{F}_2)} \leq 2^{-\binom{\lceil \log_2 |S| \rceil}{d}}. \quad (7)$$

We will now use Theorem 49 to prove the existence of low-degree dispersers for every small family of sources.

► **Corollary 50.** *Let  $n, d, k \geq 1$ , and  $\mathcal{X}$  be a family of distributions of min-entropy  $\geq k$  over  $\{0, 1\}^n$ .*

*Then a uniformly random polynomial  $p \in \mathcal{P}_2(n, d)$  is a disperser for  $\mathcal{X}$  with probability at least*

$$1 - |\mathcal{X}| \cdot 2^{1 - \binom{k}{\leq d}}.$$

**Proof.** Let  $\mathbf{X}$  be a distribution from  $\mathcal{X}$ . Since  $H_\infty(\mathbf{X}) \geq k$ , we have that  $|\text{support}(\mathbf{X})| \geq 2^k$ . By Theorem 49,

$$\Pr_{p \in_u \mathcal{P}_2(n, d)} [p|_{\text{support}(\mathbf{X})} \text{ is constant}] \leq 2^{1 - \binom{k}{\leq d}}.$$

The corollary follows by applying the union bound over all  $|\mathcal{X}|$  sources in  $\mathcal{X}$ . ◀

We will demonstrate two immediate applications of Corollary 50 for the families of local and decision forests sources.

► **Corollary 51 (Low-degree dispersers for local sources).** *Let  $1 \leq \ell \leq d \leq n$  be integers. There exists  $p \in \mathcal{P}_2(n, d)$  that is a disperser*

- *for the family of  $\ell$ -local sources on  $\{0, 1\}^n$  with min-entropy  $k > d(2^\ell n + 2\ell n \log n)^{1/d}$ .*
- *for the family of depth- $\ell$  decision forest sources on  $\{0, 1\}^n$  with min-entropy  $k > d((\ell + \log n)2^{\ell+1}n)^{1/d}$ .*

The recent result of [1] uses further properties of local sources to prove the existence of low-degree dispersers for local sources with min-entropy  $k \geq c\ell^3 d \cdot (n \log n)^{1/d}$  for a constant  $c > 0$ . Noting that every depth- $\ell$  decision forest source is also a  $(2^\ell - 1)$ -local source, the disperser of [1] for local sources implies a result similar to the above.

## 6.2 Dispersers for Polynomial and Circuit Sources

In this section, we will extend the results of the previous section to prove the existence of low-degree dispersers for powerful families of sources including polynomial-size circuits and low-degree polynomial sources. Unlike the previous examples such as local sources, the sources considered here may non-trivially depend on an arbitrary number of inputs. For example, even a degree-1 (i.e. affine) source defined by an affine map  $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$  can depend on an arbitrary number  $m \gg n$  of input bits. We get around this by restricting the map  $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$  defining the source to a low-dimensional affine subspace. Specifically, we will use the input-reduction procedure from [9], where it was used to prove that random (not necessarily bounded degree) maps extract from low-degree sources.

► **Lemma 52 ([9, Lemma 4.5]).** *Let  $m, n, k \in \mathbb{N}$ ,  $k > 1$ , and  $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$  be a function. If  $H_\infty(f(\mathbf{U}_m)) \geq k$ , then there exists an affine map  $L : \mathbb{F}_2^{11k} \rightarrow \mathbb{F}_2^m$  such that*

$$H_\infty(f(L(\mathbf{U}_{11k}))) \geq k - 1.$$

Equipped with Lemma 52, we are ready to construct dispersers for low-degree sources.

► **Theorem 53 (Low-degree disperser for lower-degree polynomial sources).** *Let  $1 \leq \ell < d \leq n$  be integers. There exists  $p \in \mathcal{P}_2(n, d)$  that is a disperser for the family of degree- $\ell$  sources on  $\{0, 1\}^n$  with min-entropy  $k \geq (12^\ell \cdot d^d \cdot n)^{\frac{1}{d-\ell}} + 1$ .*

*In particular, for every  $\ell \in \mathbb{N}$ , there is a degree- $(\ell + 2)$  disperser for degree- $\ell$  sources on  $\{0, 1\}^n$  with min-entropy  $\Omega(\sqrt{n})$ .*

► **Theorem 54** (Low-degree disperser for circuit sources). *Let  $\ell \geq 1$  and  $n \geq d \geq 2\ell + 2$  be integers. There exists  $p \in \mathcal{P}_2(n, d)$  that is a disperser for the family of  $n^\ell$ -size circuit sources on  $\{0, 1\}^n$  with min-entropy  $k \geq (30^2 \cdot d^\ell \cdot n^{2\ell})^{\frac{1}{d-2}} + 1$ .*

Theorems 53 and 54 construct low-degree dispersers for sources generated by constant-degree polynomials and polynomial-size  $\text{AC}[\oplus]$  circuits. These two classes of sources are incomparable. Indeed,  $\text{AC}[\oplus]$  computes  $\text{AND}(x_1, \dots, x_m)$  which is not a constant-degree polynomial, while constant-degree polynomials compute polynomials in  $m$  inputs which do not admit circuits of size polynomial in  $n$ . We remark that the techniques of Theorems 53 and 54 can be used to conclude the same result for a class of sources that generalizes both  $\text{AC}[\oplus]$  and constant-degree polynomials. This is the class of polynomial-size circuits which extends  $\text{AC}[\oplus]$  with gates computing arbitrary polynomials in  $m$  inputs of a fixed constant degree. For ease of exposition, we present only the results for more natural sources in Theorems 53 and 54.

## 7 Random Low-Degree Polynomials Extract from Fixed Sources

In this section, we use our bounds on the values of Hilbert functions to prove the existence of a low-degree extractor for a fixed high min-entropy source. Specifically, in Theorem 57 we show that for every source  $\mathbf{X}$  of high min-entropy, a random low-degree polynomial  $p$  has bias  $\leq \varepsilon$ , i.e.,  $\Pr_{x \in_u X}[f(x) = 1] \in 1/2 \pm \varepsilon$  with high probability. One special case of interest is the case of  $k$ -flat sources  $\mathbf{X}$  which are uniform distributions over sets of size  $2^k$ . In Section 8, we will use Theorem 57 to prove the existence of low-degree extractors for various expressive families of sources.

We start this section by using our bounds on the degree- $d$  closure of sets in order to lower-bound the probability that a random somewhat large subset  $T$  of a set  $S$  has “full Hilbert dimension”, i.e.,  $h_T(d, \mathbb{F}_2) = |T|$ . We then use this to prove Lemma 56 which states that for a large enough set  $S \subseteq \{0, 1\}^n$ , a random subset  $T \subseteq S$  of full Hilbert dimension will contain each element  $x \in S$  with almost the same probability. Finally, we present a proof of Theorem 57 which crucially relies on Lemma 56.

▷ **Claim 55.** Let  $1 \leq d \leq n$ ,  $d \leq \ell$ , and  $S \subseteq \{0, 1\}^n$ . Let  $T$  be a uniformly random subset of  $S$  of size  $\binom{\ell}{\leq d}$ . Then

$$\Pr_T \left[ h_T(d, \mathbb{F}_2) = |T| = \binom{\ell}{\leq d} \right] \geq 1 - \binom{\ell}{\leq d} \cdot 2^\ell / |S|.$$

► **Lemma 56.** Let  $1 \leq d \leq n$ ,  $d \leq \ell$ , and  $S \subseteq \{0, 1\}^n$ . Let  $T$  be a uniformly random subset of  $S$  of size  $\binom{\ell}{\leq d}$ . Then for every  $x \in S$ ,

$$(1 - \delta) \cdot \frac{\binom{\ell}{\leq d}}{|S|} \leq \Pr_T [x \in T \mid h_T(d, \mathbb{F}_2) = |T|] \leq \frac{1}{(1 - \delta)} \cdot \frac{\binom{\ell}{\leq d}}{|S|},$$

where  $\delta = \binom{\ell}{\leq d} \cdot 2^\ell / |S|$ .

Equipped with Lemma 56, we are ready to present the proof of Theorem 57.

► **Theorem 57.** Let  $n, d, k \geq 1$ , and  $\varepsilon > 0$  be a real. Then for every distribution  $\mathbf{X}$  over  $\{0, 1\}^n$  with  $H_\infty(\mathbf{X}) \geq k$ , a uniformly random degree- $d$  polynomial  $f$  is an  $\varepsilon$ -extractor for  $\mathbf{X}$ ,

$$\Pr_{x \sim \mathbf{X}} [f(x) = 1] = \frac{1}{2} \pm \varepsilon$$

with probability at least  $1 - e^{3n - \varepsilon^2 \binom{\ell}{\leq d} / (Cn^2)}$  where  $\ell = k/2 - \log(32n/\varepsilon)$  and  $C = 7 \cdot (32)^2$ .

## 8 Low-Degree Extractors

In this section, we extend the results of Section 6 to the setting of extractors. We start with the extractors version of Corollary 50 in Theorem 58, where we show that low-degree polynomials extract from small families of sources. Then, in Theorem 60, we use Theorem 58 to prove the existence of low-degree extractors for a number of families of sources. Finally, in Section 8.1, we prove the existence of low-degree extractors with multi-bit outputs.

► **Theorem 58.** *Let  $\mathcal{X}$  be a family of distributions of min-entropy  $k \geq 5 \log n$  over  $\{0, 1\}^n$  for large enough  $n$ . Let  $\mathcal{Y}$  be a family of distributions each of which is  $\varepsilon'$ -close to a convex combination of distributions from  $\mathcal{X}$ . Then for every  $d \geq 6$ , a uniformly random polynomial  $p \in \mathcal{P}_2(n, d)$  is an  $\varepsilon$ -extractor for  $\mathcal{Y}$  with probability at least*

$$1 - |\mathcal{X}| \cdot e^{3n - 30k^{d/2}/n^2}$$

for  $\varepsilon = (2d/k^{1/4})^d + \varepsilon'$ .

We will use the following input-reduction result from [9].

► **Theorem 59** ([9, Theorem 4.1]). *Let  $m, n, k \in \mathbb{N}$ ,  $k > 1$ , and  $f: \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$  be a function. If  $H_\infty(f(\mathbf{U}_m)) \geq k$ , then there exist affine maps  $L_1, \dots, L_t: \mathbb{F}_2^{11k} \rightarrow \mathbb{F}_2^m$  such that the distribution  $f(\mathbf{U}_m)$  is  $2^{-k}$ -close to a convex combination of distributions  $f(L_i(\mathbf{U}_{11k}))$ . Moreover, for each  $i \in [t]$ ,*

$$H_\infty(f(L_i(\mathbf{U}_{11k}))) \geq k - 1.$$

We are now ready to prove that low-degree polynomials extract from many sources of interest.

► **Theorem 60.** *For all  $\ell, d \geq 1$ , and all large enough  $n$ , there exists  $p \in \mathcal{P}_2(n, d)$  that is an  $\varepsilon$ -extractor for the following families of sources over  $\{0, 1\}^n$  of min-entropy  $k \geq 5 \log n$  for  $\varepsilon = 2(2d/k^{1/4})^d$ .*

- $\ell$ -local sources for  $k \geq (2^\ell n^3 \log n)^{2/d}$ .
- depth- $\ell$  decision forest sources for  $k \geq (2^\ell n^3 (\log n + \ell))^{2/d}$ .
- degree- $\ell$  sources for  $k \geq (3^\ell n)^{\frac{6}{d-2\ell}}$ .
- $n^\ell$ -size circuit sources for  $k \geq 3n^{\frac{4(\ell+1)}{d-4}}$ .

### 8.1 Extractors Outputting Multiple Bits

In Theorem 61, we show how to extend our single-bit extractors for small families of sources to the multi-bit setting, which combined with input-reduction lemma, will extend all our single-bit extractors from Theorem 60 to  $O(k)$ -bit extractors.

► **Theorem 61.** *Let  $\mathcal{X}$  be a family of distributions of min-entropy  $k \geq 5 \log n$  over  $\{0, 1\}^n$  for large enough  $n$ . Let  $\mathcal{Y}$  be a family of distributions each of which is  $\varepsilon'$ -close to a convex combination of distributions from  $\mathcal{X}$ . Then for every  $d \geq 6$  and  $t < k$ , let  $p_1, \dots, p_t \in \mathcal{P}_2(n, d)$  be independent and uniformly random polynomials. Then  $p = (p_1, \dots, p_t)$  is a  $t\varepsilon$ -extractor for  $\mathcal{Y}$  with probability at least*

$$1 - |\mathcal{X}| \cdot e^{3n+t+1-30(k-2t)^{d/2}/n^2}$$

for  $\varepsilon = (2d/k^{1/4})^d + \varepsilon'$ , assuming  $\varepsilon \leq 1/4$ .

---

**References**

---

- 1 Omar Alrabiah, Eshan Chattopadhyay, Jesse Goodman, Xin Li, and João Ribeiro. Low-degree polynomials extract from local sources. In *ICALP*, 2022.
- 2 Omar Alrabiah, Jesse Goodman, Jonathan Mosheiff, and João Ribeiro. Low-degree polynomials are good extractors. *Manuscript*, 2024.
- 3 Dave Bayer and David Mumford. What can be computed in algebraic geometry? *arXiv preprint*, 1993. [arXiv:alg-geom/9304003](https://arxiv.org/abs/alg-geom/9304003).
- 4 Paul Beame, Shayan Oveis Gharan, and Xin Yang. On the bias of Reed–Muller codes over odd prime fields. *SIAM Journal on Discrete Mathematics*, 34(2):1232–1247, 2020.
- 5 Peter Beelen and Mrinmoy Datta. Generalized Hamming weights of affine Cartesian codes. *Finite Fields and Their Applications*, 51:130–145, 2018.
- 6 Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low-degree polynomials are hard to approximate. *computational complexity*, 21(1):63–81, 2012.
- 7 Manuel Blum. Independent unbiased coin flips from a correlated biased source – A finite state Markov chain. *Combinatorica*, 6:97–108, 1986.
- 8 Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *ITCS*, 2013.
- 9 Eshan Chattopadhyay, Jesse Goodman, and Mohit Gurumukhani. Extractors for polynomial sources over  $\mathbb{F}_2$ . In *ITCS*, 2024.
- 10 Kuan Cheng and Xin Li. Randomness extraction in  $AC^0$  and with small locality. In *RANDOM*, 2018.
- 11 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing (SICOMP)*, 17(2):230–261, 1988.
- 12 Gil Cohen and Avishay Tal. Two structural results for low degree polynomials and applications. In *RANDOM*, 2015.
- 13 David Cox, John Little, and Donal O’Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2013.
- 14 Anindya De and Thomas Watson. Extractors and lower bounds for locally samplable sources. In *RANDOM*, 2011.
- 15 Yevgeniy Dodis, Shien Jin Ong, Manoj Prabhakaran, and Amit Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS*, 2004.
- 16 Yevgeniy Dodis and Kevin Ye. Doubly-affine extractors, and their applications. In *ITC*, 2021.
- 17 Dean Doron, Amnon Ta-Shma, and Roei Tell. On hitting-set generators for polynomials that vanish rarely. *Computational Complexity*, 31(2):16, 2022.
- 18 Bálint Felszeghy. *Gröbner theory of zero dimensional ideals with a view toward combinatorics*. PhD thesis, Budapest University of Technology and Economics, 2007.
- 19 Oded Goldreich, Emanuele Viola, and Avi Wigderson. On randomness extraction in  $AC^0$ . In *CCC*, 2015.
- 20 Alexander Golovnev, Zeyu Guo, Pooya Hatami, Satyajeet Nagargoje, and Chao Yan. Hilbert functions and low-degree randomness extractors, 2024. URL: <https://ecc.weizmann.ac.il/report/2024/092/>.
- 21 Petra Heijnen. *Some classes of linear codes: observations about their structure, construction, (non-)existence and decoding*. PhD thesis, Technische Universiteit Eindhoven, 1999.
- 22 Petra Heijnen and Ruud Pellikaan. Generalized Hamming weights of  $q$ -ary Reed-Muller codes. *IEEE Transactions on Information Theory (ToIT)*, 44(1):181–196, 1998.
- 23 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- 24 Xuanguo Huang, Peter Ivanov, and Emanuele Viola. Affine extractors and  $AC^0$ -parity. In *RANDOM*, 2022.
- 25 Piotr Indyk. Uncertainty principles, extractors, and explicit embeddings of  $L_2$  into  $L_1$ . In *STOC*, 2007.




- 26 Tali Kaufman, Shachar Lovett, and Ely Porat. Weight distribution and list-decoding size of Reed–Muller codes. *IEEE Transactions on Information Theory (ToIT)*, 58(5):2689–2696, 2012.
- 27 Peter Keevash and Benny Sudakov. Set systems with restricted cross-intersections and the minimum rank of inclusion matrices. *SIAM Journal on Discrete Mathematics*, 18(4):713–727, 2005.
- 28 Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for  $AC^0[\oplus]$  circuits, with applications to lower bounds and circuit compression. *Theory of Computing*, 14(12):1–24, 2018.
- 29 Jiayu Li and Tianqi Yang.  $3.1n - o(n)$  circuit lower bounds for explicit functions. In *STOC*, 2022.
- 30 Xin Li. Two source extractors for asymptotically optimal entropy, and (many) more. *arXiv preprint*, 2023. [arXiv:2303.06802](https://arxiv.org/abs/2303.06802).
- 31 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.
- 32 Shay Moran and Cyrus Rashtchian. Shattered sets and the Hilbert function. In *MFCS*, 2016.
- 33 Zipei Nie and Anthony Y. Wang. Hilbert functions and the finite degree Zariski closure in finite field combinatorial geometry. *Journal of Combinatorial Theory, Series A*, 134:196–220, 2015.
- 34 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.
- 35 Igor Carboni Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *CCC*, 2019.
- 36 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.
- 37 Zachary Remscrim. The Hilbert function, algebraic extractors, and recursive Fourier sampling. In *FOCS*, 2016.
- 38 Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences (JCSS)*, 33(1):75–87, 1986. doi:10.1016/0022-0000(86)90044-9.
- 39 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *STOC*, 1987.
- 40 Roman Smolensky. On representations by low-degree polynomials. In *FOCS*, 1993.
- 41 Srikanth Srinivasan. A robust version of Hegedüs’s lemma, with applications. *TheoretCS*, 2, 2023.
- 42 Amnon Ta-Shma and David Zucherman. Extractor codes. In *STOC*, 2001.
- 43 Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *FOCS*, 2000.
- 44 Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17:43–77, 2004.
- 45 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.
- 46 Emanuele Viola. The complexity of distributions. *SIAM Journal on Computing (SICOMP)*, 41(1):191–218, 2012.
- 47 Emanuele Viola. Extractors for circuit sources. *SIAM Journal on Computing (SICOMP)*, 43(2):655–672, 2014.
- 48 Emanuele Viola. Quadratic maps are hard to sample. *ACM Transactions on Computation Theory (TOCT)*, 8(4):1–4, 2016.
- 49 Victor K. Wei. Generalized Hamming weights for linear codes. *IEEE Transactions on Information Theory (ToIT)*, 37(5):1412–1418, 1991.

**41:24 Hilbert Functions and Low-Degree Randomness Extractors**

- 50 Avi Wigderson and David Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999. doi:10.1007/s004930050049.
- 51 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007. doi:10.4086/toc.2007.v003a006.



# Matrix Multiplication Verification Using Coding Theory\*

Huck Bennett   

University of Colorado Boulder, CO, USA

Karthik Gajulapalli   

Georgetown University, Washington DC, USA

Alexander Golovnev   

Georgetown University, Washington DC, USA

Evelyn Warton  

Oregon State University, Corvallis, OR, USA

---

## Abstract

We study the *Matrix Multiplication Verification Problem* (MMV) where the goal is, given three  $n \times n$  matrices  $A$ ,  $B$ , and  $C$  as input, to decide whether  $AB = C$ . A classic randomized algorithm by Freivalds (MFCS, 1979) solves MMV in  $\tilde{O}(n^2)$  time, and a longstanding challenge is to (partially) derandomize it while still running in faster than matrix multiplication time (i.e., in  $o(n^\omega)$  time).

To that end, we give two algorithms for MMV in the case where  $AB - C$  is *sparse*. Specifically, when  $AB - C$  has at most  $O(n^\delta)$  non-zero entries for a constant  $0 \leq \delta < 2$ , we give (1) a deterministic  $O(n^{\omega-\varepsilon})$ -time algorithm for constant  $\varepsilon = \varepsilon(\delta) > 0$ , and (2) a randomized  $\tilde{O}(n^2)$ -time algorithm using  $\delta/2 \cdot \log_2 n + O(1)$  random bits. The former algorithm is faster than the deterministic algorithm of Künnemann (ESA, 2018) when  $\delta \geq 1.056$ , and the latter algorithm uses fewer random bits than the algorithm of Kimbrel and Sinha (IPL, 1993), which runs in the same time and uses  $\log_2 n + O(1)$  random bits (in turn fewer than Freivalds’s algorithm).

Our algorithms are simple and use techniques from coding theory. Let  $H$  be a parity-check matrix of a Maximum Distance Separable (MDS) code, and let  $G = (I \mid G')$  be a generator matrix of a (possibly different) MDS code in systematic form. Our deterministic algorithm uses fast rectangular matrix multiplication to check whether  $HAB = HC$  and  $H(AB)^T = H(C^T)$ , and our randomized algorithm samples a uniformly random row  $\mathbf{g}'$  from  $G'$  and checks whether  $\mathbf{g}'AB = \mathbf{g}'C$  and  $\mathbf{g}'(AB)^T = \mathbf{g}'C^T$ .

We additionally study the *complexity* of MMV. We first show that all algorithms in a natural class of deterministic linear algebraic algorithms for MMV (including ours) require  $\Omega(n^\omega)$  time. We also show a barrier to proving a super-quadratic running time lower bound for matrix multiplication (and hence MMV) under the Strong Exponential Time Hypothesis (SETH). Finally, we study relationships between natural variants and special cases of MMV (with respect to deterministic  $\tilde{O}(n^2)$ -time reductions).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Error-correcting codes

**Keywords and phrases** Matrix Multiplication Verification, Derandomization, Sparse Matrices, Error-Correcting Codes, Hardness Barriers, Reductions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.42

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2309.16176> [7]

---

\* Due to space constraints, we have made the body of our submission a modified version of the introduction to our paper. This introduction contains a detailed overview of our work and a comparison with prior work. Nevertheless, we strongly encourage the reader to read the full version of our paper [7].



© Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn Warton; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 42; pp. 42:1–42:13



Leibniz International Proceedings in Informatics

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

**Funding** *Huck Bennett*: Supported by NSF Grant CCF-2312297.

*Karthik Gajulapalli*: Supported by NSF grant CCF-2338730.

*Alexander Golovnev*: Supported by NSF grant CCF-2338730.

**Acknowledgements** We thank Amir Nayyeri for many helpful discussions in the early stages of work on this paper, and Mark Iwen [16] for answering questions about [17]. We also thank the anonymous reviewers for their helpful comments.

## 1 Introduction

The goal of the *Matrix Multiplication Problem* (MM) is to compute the product  $AB$  of two  $n \times n$  matrices  $A$  and  $B$  given as input. Matrix multiplication has many practical and theoretical applications, and because of this has been studied by an extensive line of work. The primary goal of this work has been to determine the running time  $O(n^\omega)$  of the fastest algorithms for MM, which is captured by the matrix multiplication exponent  $\omega$ .<sup>1</sup> The best upper bounds on  $\omega$  and related quantities continue to improve [23, 3, 11, 22, 25], and [25] recently showed the current best known bound of  $\omega \leq 2.371552$ . The dream of this line of work is to show that  $\omega = 2$ , and this in fact holds under certain plausible combinatorial and group-theoretic conjectures (see [10, Conjecture 4.7 and Conjecture 3.4]). Nevertheless, showing that  $\omega = 2$  seems very challenging for the time being.

In this work, we consider a variant of matrix multiplication where the goal is to *verify* that the product of two matrices is equal to a third matrix. Specifically, we study the *Matrix Multiplication Verification Problem* (MMV) where, given three  $n \times n$  matrices  $A$ ,  $B$ , and  $C$  as input, the goal is to decide whether  $AB = C$ . MMV is clearly no harder than matrix multiplication – it can be solved in  $O(n^\omega)$  time by computing the product  $AB$  and then comparing the product entry-wise against  $C$  – but it is natural to ask whether it is possible to do better. In what became classic work, Freivalds [12] answered this question in the affirmative and gave a simple, randomized algorithm that solves MMV in  $\tilde{O}(n^2)$  time. This  $\tilde{O}(n^2)$  running time bound is essentially the best possible, and so, unlike matrix multiplication, the complexity of MMV is relatively well understood.

However, it is in turn natural to ask whether it is possible to *derandomize* Freivalds’s algorithm partially or completely. More specifically, it is natural to ask whether it is possible to give a *deterministic* algorithm for MMV running in  $\tilde{O}(n^2)$  time or at least  $O(n^{\omega-\varepsilon})$  time for constant  $\varepsilon > 0$ .<sup>2</sup> Or, if it is not possible to give a deterministic algorithm for MMV with these running times, it is natural to ask whether it is possible to use fewer random bits than Freivalds’s algorithm, which uses  $n$  random bits. Trying to answer these questions has become a key goal for derandomization efforts, and has received substantial study [4, 19, 20, 21].

<sup>1</sup> Formally,  $\omega$  is defined as the infimum over  $\omega'$  such that the product of two  $n \times n$  matrices can be computed in  $O(n^{\omega'})$  time. So, MM algorithms are actually only guaranteed to run in  $O(n^{\omega+\varepsilon})$  time for any constant  $\varepsilon > 0$ .

<sup>2</sup> We use the notation  $\tilde{O}(f(n))$  to mean  $f(n) \cdot \text{poly}(\log f(n))$ . Freivalds’s algorithm uses  $O(n^2)$  arithmetic operations, each of which takes  $\text{poly}(\log n)$  time when working over integer matrices with entries bounded in magnitude by  $\text{poly}(n)$ ; we assume this setting in the introduction.

Of course, it is only possible for such a  $O(n^{\omega-\varepsilon})$ -time algorithm to exist if  $\omega > 2$ . We assume that this is the case throughout the introduction.

■ **Table 1** Algorithms for MMV on matrices  $A, B, C \in \mathbb{Z}^{n \times n}$  with entries of magnitude at most  $\text{poly}(n)$  and such that  $AB - C$  has at most  $n^\delta$  non-zero entries for  $0 \leq \delta \leq 2$ . Our new algorithms are shown in bold. We list asymptotic running times, with  $\text{poly}(\log n)$  factors suppressed for readability, and the number of random bits used to achieve success probability  $1/2$ . (Each of the three listed randomized algorithms has one-sided error, so this probability is meaningful.) Here  $\omega(\cdot, \cdot, \cdot)$  is the rectangular matrix multiplication exponent,  $\omega = \omega(1, 1, 1)$  is the (square) matrix multiplication exponent, and  $\varepsilon > 0$  is an arbitrarily small positive constant.

| Algorithm                             | Asymptotic Runtime                     | Bits of Randomness                        |
|---------------------------------------|--|---|
| Matrix Multiplication                 | $n^{\omega+\varepsilon}$               | 0   |
| Random Entry Sampling (folklore)      | $n^{3-\delta}$                         | $2n^{2-\delta} \cdot \log_2(n) + O(1)$    |
| Freivalds's Algorithm [12]            | $n^2$                                  | $n$                                       |
| Vandermonde Mat. Sampling [19]        | $n^2$                                  | $\log_2(n) + O(1)$                        |
| Multipoint Poly. Evaluation [21]      | $n^2 + n^{1+\delta}$                   | 0   |
| Cauchy Bound [20]                     | $n^3$ ( $n^2$ in Integer RAM)          | 0   |
| <b>Parity Check/Fast RMM (Thm. 1)</b> | $n^{\omega(1,1,\delta/2)+\varepsilon}$ | 0   |
| <b>Cauchy Mat. Sampling (Thm. 2)</b>  | $n^2$                                  | $\frac{\delta}{2} \cdot \log_2(n) + O(1)$ |

## 1.1 Our Results

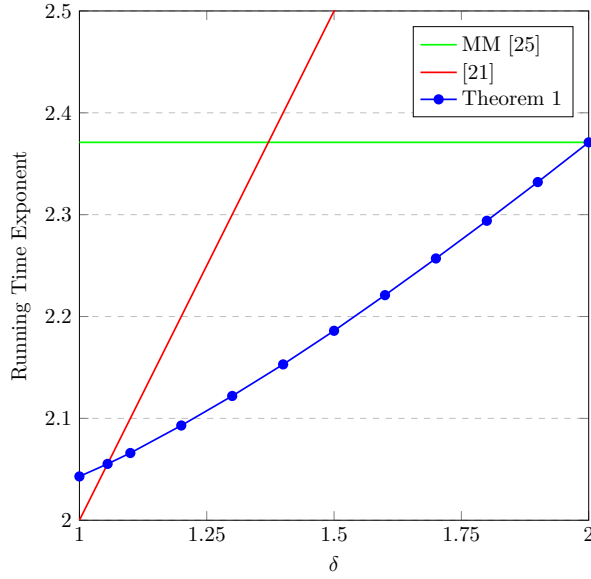
Our main results are two new algorithms for the Matrix Multiplication Verification Problem in the *sparse* regime, i.e., in the case where  $AB - C$  is promised to have few non-zero entries (if any). See Table 1 for a summary of our algorithms and how they compare to other known algorithms for MMV. Additionally, we give a barrier for giving a fast algorithm for MMV using a broad class of linear algebraic techniques, a barrier to showing hardness of MMV, and reductions between variants of MMV.

### 1.1.1 Algorithms

Besides being inherently interesting, MMV in the sparse regime is the natural decision version of the well-studied *Output-Sensitive Matrix Multiplication Problem* (OSMM). It is also motivated by the following scenario. Suppose that Alice wants to compute the product  $AB$  of two large matrices  $A$  and  $B$ , but has restricted computational resources. So, she sends  $A$  and  $B$  to Bob, who has more extensive computational resources. Bob computes the product  $AB$ , and sends the result back to Alice over a noisy channel (without error-correction, which increases the size of the message), from which Alice receives a matrix  $C$ . Alice knows that either  $C = AB$  as desired, or that  $C$  is corrupted but (with high probability) only differs from  $AB$  in a few entries. She wants to check which is the case efficiently.

We define  $\|\mathbf{v}\|_0$  (respectively,  $\|M\|_0$ ) to be the number of non-zero entries in (i.e., Hamming weight of) a vector  $\mathbf{v}$  (respectively, matrix  $M$ ). We call a vector  $\mathbf{v}$  (respectively, matrix  $M$ ) *t-sparse* if  $\|\mathbf{v}\|_0 \leq t$  (respectively, if  $\|M\|_0 \leq t$ ).

Our first algorithm (given in Figure 2) is deterministic, and uses fast rectangular matrix multiplication. For  $\alpha, \beta, \gamma \in [0, 1]$ , let the rectangular matrix multiplication exponent  $\omega(\alpha, \beta, \gamma)$  be the infimum over values  $\omega' > 0$  such that the product of a  $n^\alpha \times n^\beta$  matrix and a  $n^\beta \times n^\gamma$  matrix can be computed using  $O(n^{\omega'})$  arithmetic operations. Note that  $\omega = \omega(1, 1, 1)$  is the standard (square) matrix multiplication exponent.



■ **Figure 1** Running times of deterministic algorithms for MMV when  $AB - C$  is  $O(n^\delta)$ -sparse for  $1 \leq \delta \leq 2$ . Our algorithm from Theorem 1 is faster than the best known algorithms for matrix multiplication [25] and faster than Künnemann’s algorithm [21] for all  $1.056 \leq \delta < 2$ . The plotted blue points corresponding to the running time of the algorithm in Theorem 1 are derived from the bounds on  $\omega(1, 1, \delta/2)$  in [25, Table 1]. The line segments connecting them are justified by the fact that  $\omega(1, 1, \cdot)$  is a convex function.

- (1) Let  $k := \lceil \sqrt{t} \rceil$ .
- (2) Compute an arbitrary prime  $p$  that satisfies  $n \leq p \leq 2n$ .
- (3) Compute a (parity check) matrix  $H \in \mathbb{F}_p^{k \times n}$  of a code with distance at least  $k + 1$ .
- (4) Output YES if  $H(AB - C) = 0$  and  $H(AB - C)^T = 0$  (where arithmetic is performed over  $\mathbb{Z}$ ), and output NO otherwise.

■ **Figure 2** Deterministic Algorithm for MMV corresponding to Theorem 1.

► **Theorem 1** (Fast deterministic MMV for sparse matrices, informal). *Let  $A, B, C \in \mathbb{Z}^{n \times n}$  be matrices satisfying  $\max_{i,j} \{|A_{i,j}|, |B_{i,j}|, |C_{i,j}|\} \leq n^c$  for some constant  $c > 0$  and satisfying  $\|AB - C\|_0 \leq n^\delta$  for  $0 \leq \delta \leq 2$ . Then for any constant  $\varepsilon > 0$ , there is a deterministic algorithm for MMV on input  $A, B, C$  that runs in  $O(n^{\omega(1,1,\delta/2)+\varepsilon})$  time.*

We note that  $\omega(1, 1, \beta) < \omega$  for all  $\beta < 1$  (assuming  $\omega > 2$ );<sup>3</sup> and so our algorithm is faster than matrix multiplication when  $AB - C$  is promised to be  $O(n^\delta)$ -sparse for constant  $\delta < 2$ . Furthermore, it is faster than Künnemann’s algorithm [21], which is also for MMV in the regime where  $AB - C$  is sparse, when  $\omega(1, 1, \delta/2) < 1 + \delta$ . The equation  $\omega(1, 1, \delta/2) = 1 + \delta$  whose unique solution corresponds to the crossover point at which our algorithm becomes faster than Künnemann’s turns out to be relevant in other contexts too [27], and [23, 25] both provide bounds on its solution. Specifically, [25] shows that the solution  $\delta$  to this equation satisfies  $\delta \leq 1.056$ , and so our algorithm in Theorem 1 is (strictly) faster than any previously known deterministic algorithm for MMV when  $1.056 \leq \delta < 2$ .

<sup>3</sup> See [7, Theorem 2.3]

- (1) Let  $k := \lceil \sqrt{t}/\varepsilon \rceil$ .
- (2) Compute an arbitrary prime  $p$  satisfying  $k + n \leq p < 2(k + n)$ .
- (3) Compute a  $\lceil \sqrt{t} \rceil$ -regular matrix  $S = (\mathbf{s}_1, \dots, \mathbf{s}_k) \in \mathbb{F}_p^{n \times k}$ .
- (4) Sample a uniformly random column index  $i \sim \{1, \dots, k\}$ .
- (5) Output YES if  $AB\mathbf{s}_i = C\mathbf{s}_i$  and  $(AB)^T \mathbf{s}_i = C^T \mathbf{s}_i$  (where arithmetic is performed over  $\mathbb{Z}$ ), and output NO otherwise.

■ **Figure 3** Randomized Algorithm for MMV corresponding to Theorem 2.

Additional bounds on  $\omega(1, \delta/2, 1) = \omega(1, 1, \delta/2)$  – and hence the running time of the algorithm in Theorem 1 – appear in [25, Table 1]. For example, that table shows that  $\omega(1, 1, 0.55) < 2.067$  and  $\omega(1, 1, 0.95) < 2.333$  (which correspond to  $\delta = 1.1$  and  $\delta = 1.9$ , respectively). We also note that our algorithm runs in essentially optimal  $\tilde{O}(n^2)$  time when  $\delta \leq 0.642 \leq 2\omega^\perp$ , where  $\omega^\perp := \sup\{\omega' > 0 : \omega(1, 1, \omega') = 2\} \geq 0.321$  is the dual matrix multiplication exponent [25], but that Künnemann’s algorithm [21] runs in  $\tilde{O}(n^2)$  time for any  $\delta \leq 1$ .

Our second algorithm runs in  $\tilde{O}(n^2)$  time, but is randomized (see Figure 3). It uses few bits of randomness when  $AB - C$  is sparse.

► **Theorem 2** (Fast randomized MMV for sparse matrices, informal). *Let  $c > 0$  be a constant, let  $A, B, C \in \mathbb{Z}^{n \times n}$  be matrices satisfying  $\max_{i,j} \{|A_{i,j}|, |B_{i,j}|, |C_{i,j}|\} \leq n^c$  and satisfying  $\|AB - C\|_0 \leq n^\delta$  for  $0 \leq \delta \leq 2$ , and let  $\varepsilon = \varepsilon(n) \geq 1/n$ . Then there is a randomized algorithm for MMV on input  $A, B, C$  that runs in  $\tilde{O}(n^2)$  time, succeeds with probability  $1 - \varepsilon$ , and uses at most  $\lceil \delta/2 \cdot \log_2(n) + \log_2(1/\varepsilon) \rceil$  bits of randomness.*

Theorem 2 improves on the number of random bits used by the algorithm of Kimbrel and Sinha [19] when  $\delta < 2$  (which uses  $\log_2(n) + \log_2(1/\varepsilon) + O(1)$  random bits regardless of the sparsity of  $AB - C$ ), and matches the number of random bits used by their algorithm when  $\delta = 2$ . The algorithms both run in  $\tilde{O}(n^2)$  time. In fact, one may think of the algorithm summarized in Theorem 2 as a natural extension of the algorithm in [19] to handle the sparse case more efficiently, although it requires additional techniques to implement. (Our algorithm requires matrices with a stronger pseudorandomness property than theirs; see the “algorithmic techniques” section below.)

We note that Theorem 2 only improves on known algorithms when  $1 < \delta < 2$ , and only by a factor of  $\delta/2$ . Indeed, as mentioned above, when  $\delta \leq 1$  Künnemann’s algorithm [21] solves MMV *deterministically* in  $\tilde{O}(n^2)$  time, and when  $\delta = 2$  our algorithm matches the number of random bits used by Kimbrel and Sinha’s algorithm. Although seemingly modest, this constant-factor improvement is not surprising: any super-constant improvement on the number of bits used by [19] (i.e., an MMV algorithm using  $o(\log n)$  random bits) could be turned into a deterministic algorithm for MMV with only a sub-polynomial (i.e.,  $n^{o(1)}$ ) multiplicative increase in running time.

### 1.1.1.1 Algorithmic techniques

Here we briefly summarize the techniques that we use for the MMV algorithms corresponding to Theorems 1 and 2. We start by remarking that Theorems 1 and 2 hold not just for matrices over  $\mathbb{Z}$  with entries of polynomial magnitude, but also for matrices over all finite

fields  $\mathbb{F}_q$  with  $q \leq \text{poly}(n)$ .<sup>4</sup> In fact, our algorithms work “natively” in the finite field setting – i.e., on  $n \times n$  matrices  $A, B, C$  over finite fields  $\mathbb{F}_q$  – which is directly amenable to using techniques from coding theory. We assume this setting in the description below. Furthermore, there is a linear-time, sparsity-preserving reduction from MMV to the special case of MMV where  $C$  is fixed as  $C = 0$  and the goal is to decide whether  $AB = 0$  for input matrices  $A, B$  (see [21, Proposition 3.1]). We will also generally assume this setting in the introduction.

For both algorithms, we will use the observation that if  $AB - C$  is non-zero and  $t$ -sparse then at least one row or column of  $AB - C$  must be non-zero and  $k$ -sparse for  $k := \lfloor \sqrt{t} \rfloor$ . A similar observation appears in [26].

Our first, deterministic algorithm (Theorem 1) uses a matrix  $H$  over  $\mathbb{F}_q$  such that any  $k$  columns of  $H$  are linearly independent. Equivalently, we require a matrix  $H \in \mathbb{F}_q^{m \times n}$  such that for all non-zero vectors  $\mathbf{x} \in \mathbb{F}_q^n$  with  $\|\mathbf{x}\|_0 \leq k$  (corresponding to a sparse, non-zero column or row of  $AB$ ),  $H\mathbf{x} \neq \mathbf{0}$ . This is exactly the property guaranteed by the parity-check matrix  $H$  of an error correcting code  $\mathcal{C} \subseteq \mathbb{F}_q^n$  with minimum distance  $d > k$ . Moreover, if a code  $\mathcal{C}$  with minimum distance  $d = k + 1$  is a so-called Maximum Distance Separable (MDS) code, then it has a  $k \times n$  parity-check matrix  $H$ . MDS codes with useful parameters exist and have efficiently constructible parity-check matrices. In particular, (generalized) Reed-Solomon codes are MDS codes, and exist when  $k \leq n \leq q$  (see, e.g., [14]). Their parity-check matrices  $H$  are Vandermonde matrices, which are constructible in  $kn \cdot \text{poly}(\log q) \leq n^2 \cdot \text{poly}(\log q)$  time.

Our algorithm then uses fast rectangular matrix multiplication to compute  $HAB = (HA)B$  and  $H(AB)^T = (HB^T)A^T$  using roughly  $n^{\omega(1,1,\delta/2)}$  arithmetic operations, where  $0 \leq \delta \leq 2$  is such that  $t \leq n^\delta$ . If  $AB = 0$ , then  $HAB = H(AB)^T = 0$ . On the other hand, if  $AB \neq 0$  then  $AB$  is  $t$ -sparse and therefore has a  $k$ -sparse row or column. So, at least one of the expressions  $HAB$  and  $H(AB)^T$  is non-zero.

Our second, randomized algorithm (Theorem 2) uses a matrix  $S \in \mathbb{F}_q^{m \times n}$  with the property that all of its  $k \times k$  submatrices are non-singular. Matrices  $S$  with this property are called  $k$ -regular, and matrices  $S$  all of whose square submatrices (of any size) are non-singular are called *super regular*.<sup>5</sup> We note that  $k$ -regularity is stronger than the property we require for  $H$  in the first algorithm. In particular, if a matrix  $S \in \mathbb{F}_q^{m \times n}$  is  $k$ -regular and  $0 < \|\mathbf{x}\|_0 \leq k$ , then  $\|S\mathbf{x}\|_0 \geq m - k + 1$ . I.e.,  $S$  being  $k$ -regular implies not only that  $S\mathbf{x}$  is non-zero, but that  $S\mathbf{x}$  has relatively high Hamming weight for such  $\mathbf{x}$ . This property is useful because it implies that  $\Pr[\langle \mathbf{s}, \mathbf{x} \rangle \neq 0] \geq (m - k + 1)/m$ , where  $\mathbf{s}$  is a random row of  $S$ . Indeed, this observation leads to our second algorithm: we sample a random row  $\mathbf{s}$  from a  $k$ -regular matrix  $S \in \mathbb{F}_q^{m \times n}$  and check whether  $\mathbf{s}AB = \mathbf{0}$  and  $\mathbf{s}(AB)^T = \mathbf{0}$ . Setting, e.g.,  $m = 2k$ , we get that this algorithm succeeds with probability at least  $(2k - k + 1)/(2k) > 1/2$ .

It remains to construct (rows of)  $k$ -regular matrices  $S$  efficiently. Although a priori it is not even obvious that  $k$ -regular matrices exist for arbitrary  $k$ , in fact *super regular* matrices exist and are efficiently constructible. Specifically, we use a family of super regular (and hence  $k$ -regular) matrices called *Cauchy matrices*; the entries of such a matrix  $S$  are defined as  $S_{i,j} = 1/(x_i - y_j)$ , where  $x_1, \dots, x_m, y_1, \dots, y_n$  are distinct elements of  $\mathbb{F}_q$ . In fact, as follows from their definition, given a (random) index  $1 \leq i \leq m$ , it is even possible to construct the  $i$ th row of a Cauchy matrix  $S$  efficiently without computing the rest of the matrix, as needed.

<sup>4</sup> The algorithms also work over larger finite fields, but with slower running times due to the increased bit complexity of performing arithmetic operations over those fields.

<sup>5</sup> For formal definitions see [7, Section 2.4]

Finally, we remark that there is a deep connection between MDS codes and super regular matrices (and between generalized Reed-Solomon codes, Vandermonde matrices, and Cauchy matrices). Specifically, if  $G = (I \mid S)$  is the generator matrix of an MDS code in systematic form, then  $S$  is a super regular matrix [24]. Moreover, if such a matrix  $G$  is the generator matrix of a generalized Reed-Solomon code, then  $S$  is a Cauchy matrix [24].

### 1.1.2 Barriers

The dream for the line of work described in this paper is to give a deterministic,  $\tilde{O}(n^2)$ -time algorithm for MMV on arbitrary matrices. However, achieving this goal has proven to be very difficult despite a substantial amount of work towards it. So, it is natural to ask whether perhaps no such algorithm exists, i.e., whether MMV is in some sense *hard*. We first show a result in this direction, and then show a barrier result to showing SETH hardness of MMV (and even MM).<sup>6</sup>

#### 1.1.2.1 Linear algebraic algorithms barrier

We first prove that a natural class of deterministic linear algebraic algorithms for MMV based on multiplying smaller matrices – including the algorithm in Theorem 1 – cannot run in less than  $n^\omega$  time when using a matrix multiplication subroutine running in worst-case rectangular matrix multiplication time and when performing all multiplications independently. Specifically, the  $\Omega(n^\omega)$  lower bound holds if for all  $\alpha, \beta \geq 0$ , the subroutine requires  $\Omega(n^{\omega(1,1,\alpha)})$  to compute the product of an  $n \times n^\alpha$  matrix and an  $n \times n$ , and  $\Omega(n^{\omega(1,1,\beta)})$  time to compute the product of an  $n \times n$  matrix and an  $n \times n^\beta$  matrix.

The idea is that natural algorithms for verifying that  $AB = C$  for  $n \times n$  matrices  $A, B, C$  including ours amount to performing  $k$  “zero tests.” More specifically, the  $i$ th such test checks that  $L_i(AB - C)R_i = 0$  for some fixed  $n^{\alpha_i} \times n$  matrix  $L_i$  and  $n \times n^{\beta_i}$  matrix  $R_i$ , where  $\alpha_i, \beta_i \in [0, 1]$ . We observe that the conditions  $L_i(AB - C)R_i = 0$  for  $i = 1, \dots, k$  together correspond to a homogeneous system of  $\sum_{i=1}^k n^{\alpha_i + \beta_i}$  linear equations in the  $n^2$  variables corresponding to the entries of  $X = AB - C$  for  $1 \leq i, j \leq n$ . So, for this system to have  $X_{i,j} = 0$  for  $1 \leq i, j \leq n$  as its unique solution, it must be the case that  $\sum_{i=1}^k n^{\alpha_i + \beta_i} \geq n^2$ , which we show implies that  $\sum_{i=1}^k n^{\omega(1,1,\min(\alpha_i,\beta_i))} \geq \sum_{i=1}^k n^{\omega(\alpha_i,1,\beta_i)} \geq n^\omega$ . Therefore, an algorithm that independently computes each product  $L_i A B R_i$  in time  $\Omega(n^{\omega(1,1,\min(\alpha_i,\beta_i))})$  uses  $\Omega(n^\omega)$ .<sup>7</sup>

#### 1.1.2.2 A barrier to SETH-hardness of MM

While under certain reasonable conjectures, the matrix multiplication exponent  $\omega = 2$  (see [10, Conjecture 4.7 and Conjecture 3.4]), the best provable upper bound we have is  $\omega < 2.371552$  by [25]. Nevertheless, given the apparent difficulty of showing  $\omega \approx 2$ , it is natural to ask whether MM is in fact *hard*. To that end, we study showing its hardness under the *Strong Exponential Time Hypothesis* (SETH). However, rather than showing SETH-hardness of MM, we show a *barrier* to proving  $n^\gamma$ -hardness of MM for constant  $\gamma > 2$  under SETH. (Because MMV is trivially reducible to MM, our hardness barrier result also applies to MMV.)

<sup>6</sup> More properly, our first result is a barrier to giving a fast algorithm for MMV, and our second result is a barrier to showing hardness of MMV (i.e., it “gives a barrier to giving a barrier” for a fast MMV algorithm).

<sup>7</sup> For a more detailed exposition of this barrier see [7, Section 3.3]

We informally define several concepts used in the statement of our result. SETH says that for large constant  $k$ ,  $k$ -SAT instances on  $n$  variables take nearly  $2^n$  time to solve, and the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH) says that certifying that such  $k$ -SAT formulas are *not* satisfiable takes nearly  $2^n$  time even for nondeterministic algorithms. We call a matrix *rigid* if the Hamming distance between it and all low-rank matrices is high (the Hamming distance and rank are quantified by two parameters). Rigid matrices have many connections to complexity theory and other areas, and a key goal is to find explicit, deterministic constructions of such matrices.

Intuitively, NSETH rules out showing hardness of problems with non-trivial co-nondeterministic algorithms under SETH. Somewhat more precisely, assuming NSETH, problems contained in  $\text{coTIME}[f(n)]$  (but perhaps only known to be in  $\text{TIME}[g(n)]$  for  $g(n) = \omega(f(n))$ ), cannot be shown to be  $\Omega(f(n)^{1+\varepsilon})$ -hard under SETH.<sup>8</sup> Künnemann [21] noted that, because Freivald’s algorithm shows that MMV is in  $\text{coTIME}[n^2 \cdot \text{poly}(\log n)]$ , NSETH rules out showing  $\Omega(n^\gamma)$  hardness of MMV under SETH for constant  $\gamma > 2$ .

In this work, we extend this observation and give a barrier not only to showing SETH-hardness of MMV but to showing hardness of MM. We demonstrate that, if there exists a constant  $\gamma > 2$  and a reduction from  $k$ -SAT to MM such that a  $O(n^{\gamma-\varepsilon})$ -time algorithm for MM for any constant  $\varepsilon > 0$  breaks SETH, then either (1) the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH) is false, or (2) a new non-randomized algorithm for computing (arbitrarily large) rigid matrices exists. We also note that, by known results, falsifying NSETH implies a new circuit lower bound as a consequence. In short, our barrier result says that showing  $n^\gamma$ -hardness of MM under SETH for  $\gamma > 2$  would lead to major progress on important questions in complexity theory.<sup>9</sup>

A key idea that we use for proving our result is that it is possible to compute the product of two *non-rigid* matrices efficiently using a *nondeterministic* algorithm. This follows from two facts. First, by definition, a non-rigid matrix is the sum of a low-rank matrix  $L$  and a sparse matrix  $S$ , and using nondeterminism it is possible to guess  $L$  and  $S$  efficiently. Second, it is possible to compute the product of two sparse matrices or a low-rank matrix and another matrix efficiently. (In fact, we also use nondeterminism to guess a *rank factorization* of  $L$ , and this factorization is what allows for fast multiplication by  $L$ .)

Very roughly, we prove the barrier result as follows. We first suppose that there is a reduction from  $k$ -SAT to (potentially multiple instances of) matrix multiplication. In particular, such a reduction outputs several pairs of matrices to be multiplied. We then analyze three cases:

1. If the matrices output by this reduction always have small dimension (as a function of  $n$ ), then we can compute the product of each pair quickly using standard matrix multiplication algorithms (even using naïve, cubic-time matrix multiplication). This leads to a fast, deterministic algorithm for  $k$ -SAT, which refutes SETH (and hence NSETH).
2. If the matrices output by this reduction are always *not* rigid, then we can compute the product of each pair quickly using the nondeterministic algorithm sketched above. This leads to a fast, nondeterministic algorithm for showing that  $k$ -SAT formulas are not satisfiable, which refutes NSETH.
3. Finally, if neither of the above cases holds, then the reduction must sometimes output rigid matrices with large dimension as a function of  $n$ . So, we obtain an algorithm for generating arbitrarily large rigid matrices using an NP oracle: iterate through all  $k$ -SAT

<sup>8</sup> We refer the reader to see [7, Definition 2.22] for a formal definition of SETH-hardness.

<sup>9</sup> See see [7, Section 4] for a more detailed exposition.



formulas  $\varphi$  with at most a given number of variables, apply the reduction from  $k$ -SAT to MM to each formula, and then use the NP oracle to check whether each large matrix output by the reduction is rigid.

We remark that although NSETH is a strong and not necessarily widely believed conjecture, [18, 9] showed that refuting it (as in Item 2 above) would nevertheless imply an interesting new circuit lower bound. Specifically, they showed that if NSETH is false, then the complexity class  $E^{NP}$  requires series-parallel circuits of size  $\omega(n)$ .

Additionally, we remark that despite how slow the “iterate through all sufficiently large  $k$ -SAT formulas and apply the  $k$ -SAT-to-MM reduction to each one” algorithm described in Item 3 seems, it would still substantially improve on state-of-the-art non-randomized algorithms for generating rigid matrices. This is also true despite the fact that the algorithm uses an NP oracle.<sup>10</sup>

### 1.1.3 Reductions

Again, motivated by the apparent challenge of fully derandomizing Freivalds’s algorithm, we study relationships between variants of MMV with the goal of understanding what makes the problem hard to solve deterministically in  $\tilde{O}(n^2)$  time but easy to solve in  $\tilde{O}(n^2)$  time using randomness (in contrast to MM). More specifically, we study which variants are potentially easier than MMV (i.e., reducible to MMV, but not obviously solvable deterministically in  $\tilde{O}(n^2)$  time using known techniques), equivalent to MMV, and potentially harder than MMV (i.e., variants to which MMV is reducible, but which are not obviously as hard as MM). We study these questions by looking at deterministic  $\tilde{O}(n^2)$ -time reductions between variants. See Figure 4 for a summary of our results.

First, we show that two apparently special cases of MMV are in fact equivalent to MMV. These special cases are: (1) the *Inverse Verification Problem*, where the goal is to verify that  $B = A^{-1}$  for input matrices  $A$  and  $B$  (equivalently, the special case of MMV where  $C = I_n$ ), and (2) the *Symmetric MMV Problem*, where the input matrices  $A$  and  $B$  (but not necessarily  $C$ ) are symmetric.<sup>11</sup> These reductions are relatively simple, and complement the (also simple) reduction of [21], who showed that the All Zeroes Problem (i.e., the special case of MMV where  $C = 0$ ) is MMV-complete.

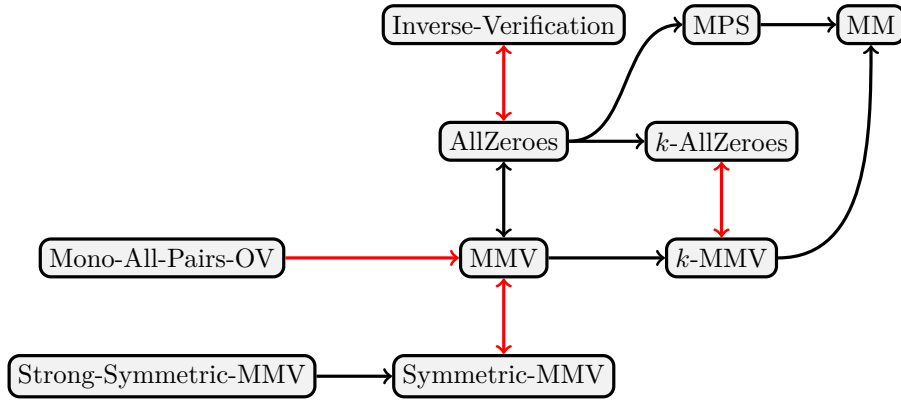
Second, we identify two problems that are  $\tilde{O}(n^2)$ -time reducible to MMV, but are not clearly solvable in  $\tilde{O}(n^2)$  time or equivalent to MMV. These problems are: (1) the *Strong Symmetric MMV Problem*, where all three of the input matrices  $A$ ,  $B$ , and  $C$  are symmetric, and (2) the Monochromatic All Pairs Orthogonal Vectors Problem, where the goal is, given vectors  $\mathbf{a}_1, \dots, \mathbf{a}_n$  to decide whether  $\langle \mathbf{a}_i, \mathbf{a}_j \rangle = 0$  for all  $i \neq j$ .

Third, we identify two problems for which there are  $\tilde{O}(n^2)$ -time reductions from MMV and that are  $\tilde{O}(n^2)$ -time reducible to MM. These “MMV/MM-intermediate problems” are: (1) the Matrix Product Sparsity Problem (MPS), in which the goal is, given matrices  $A$  and  $B$  and  $r \geq 0$  as input, to decide whether  $\|AB\|_0 \leq r$ , and (2) the  $k$ -MMV problem, in which given matrices  $A_1, \dots, A_k, C$  as input, the goal is to decide whether  $\prod_{i=1}^k A_i = C$ . We note that MPS is equivalent to the counting version of the Orthogonal Vectors Problem ( $\#OV$ ).<sup>12</sup> We additionally show that  $k$ -MMV is equivalent to the  $k$ -All Zeroes problem,

<sup>10</sup> See [7, Section 4] for a more thorough discussion.

<sup>11</sup> See [7, Section 5] for the complete reductions

<sup>12</sup> Indeed, Monochromatic All Pairs Orthogonal Vectors is no harder than MMV (and not obviously equivalent), Bichromatic All Pairs Orthogonal Vectors is equivalent to the All Zeroes Problem and is



■ **Figure 4** A diagram of reductions among MMV and related problems on  $n \times n$  matrices. Arrows represent deterministic  $O(n^2)$ -time reductions (and double-headed arrows denote equivalences under such reductions). Red arrows indicate new (non-trivial) reductions.

i.e.,  $k$ -MMV where  $C$  is fixed to be 0. See Figure 4 for a summary of these variants and reductions between them. For a full presentation of definitions and reductions we refer the reader to [7, Section 5].

## 1.2 Related Work

We next discuss other algorithms for MMV and related problems on  $n \times n$  integer matrices  $A$ ,  $B$ , and  $C$ . We summarize these algorithms, as well as ours, in Table 1. We start by noting that it suffices to consider the special case of MMV where  $C = 0$  (i.e., where the goal is to decide whether  $AB = 0$ ), which is called the All Zeroes Problem. Indeed, a result from [21] shows that there is a simple  $O(n^2)$ -time reduction from MMV on  $n \times n$  matrices  $A, B, C$  to the All Zeroes problem on  $2n \times 2n$  matrices  $A', B'$  with the property that  $\|AB - C\|_0 = \|A'B'\|_0$ . So, for this section we consider the All Zeroes Problem without loss of generality.

Perhaps the most closely related works to ours are [17, 1], which use fast rectangular matrix multiplication for the *Output-Sensitive Matrix Multiplication Problem* (OSMM). In  $t$ -OSMM, the goal is, given matrices  $A, B$  as input, to compute the product  $AB$  when it is promised to be  $t$ -sparse. There is a trivial reduction from MMV when the output is promised to be  $t$ -sparse to  $t$ -OSMM – compute  $AB$  and check whether it is equal to 0. Indeed, OSMM is essentially the search version of sparse MMV. In particular, [17] use techniques from compressive sensing to solve OSMM. However, it is not clear that the measurement matrix  $M$  in [17] can be constructed deterministically in  $\tilde{O}(n^2)$  time, and so the algorithm in [17] is a non-uniform algorithm as described. There are other candidate measurement matrices with deterministic constructions that may work for a similar purpose [16], but the exact tradeoffs do not seem to have been analyzed and it is not clear that it is possible to get a (uniform) algorithm with the same parameters. Additionally, [17] only handles the case when all columns or rows of  $AB$  are promised to have a given sparsity, rather than the case where there is a “global bound” of  $t$  on the sparsity of the matrix product itself.

---

therefore equivalent to MMV, and MPS/#OV is at least as hard as MMV. In the fine-grained complexity setting OV variants are usually considered with  $n$  vectors in dimension  $d = \text{poly}(\log n)$ ; here we are considering the regime where  $d = n$ .

The main algorithm in [1] for OSMM (summarized in [1, Theorem 1.4]) runs in randomized time  $O(n^{1.3459\delta})$  when both the input matrices  $A, B$  and their product  $AB$  are  $n^\delta$ -sparse.<sup>13</sup> For the special case when all entries in  $A, B$  are non-negative [1] give a deterministic algorithm with the same running time as their randomized algorithm. We note that [1] was written independently and concurrently with this work.

Besides simply using matrix multiplication, perhaps the most natural idea for an algorithm for the All Zeros problem is to compute a random entry  $(AB)_{i,j}$  of  $AB$  and check whether it is non-zero. If  $\|AB\|_0 \geq n^\delta$ , then sampling, say,  $10n^{2-\delta}$  random entries of  $AB$  independently will find a non-zero entry with good constant probability. Because computing each such entry amounts to computing an inner product, and sampling indices  $i, j \sim \{1, \dots, n\}$  takes roughly  $2 \log_2 n$  random bits, this algorithm overall takes  $\tilde{O}(n^{3-\delta})$  time and  $O(n^{2-\delta} \log n)$  random bits. So, this algorithm is relatively efficient and succeeds with good probability in the case when  $AB$  is dense, but even then requires a relatively large number of random bits. We also note the somewhat odd fact that this algorithm is most efficient when  $AB$  is dense, whereas our algorithms are most efficient when  $AB$  is sparse.

Freivalds’s algorithm [12] works by sampling a uniformly random vector  $\mathbf{x} \sim \{0, 1\}^n$ , and outputting “YES” if  $AB\mathbf{x} = \mathbf{0}$  and “NO” otherwise. If  $AB = \mathbf{0}$ , then this algorithm is always correct, and if  $AB \neq \mathbf{0}$  then it fails with probability at most  $1/2$ .<sup>14</sup> In particular, Freivalds’s algorithm has one-sided error with no false negatives (i.e., it is a **coRP** algorithm).

A key idea for subsequent algorithms was to reduce MMV to a question about polynomials. The main idea is the following. Define  $\mathbf{x} := (1, x, x^2, \dots, x^{n-1})^T$ , where  $x$  is an indeterminate, and define  $p_i(x) := (AB\mathbf{x})_i = \sum_{j=1}^n (AB)_{i,j} \cdot x^{j-1}$ . Note that  $AB = \mathbf{0}$  if and only if the polynomials  $p_i(x)$  are identically zero (as formal polynomials) for all  $i \in \{1, \dots, n\}$ . Furthermore, if the  $i$ th row of  $AB$  is non-zero then  $p_i(x)$  is a non-zero polynomial of degree at most  $n - 1$ , and therefore has at most  $n - 1$  distinct complex (and hence integral) roots. So, for such  $p_i(x)$  and a non-empty set  $S \subset \mathbb{Z}$ ,  $\Pr_{\alpha \sim S}[p(\alpha) = 0] \leq (n - 1)/|S|$ , which is less than  $1/2$  when  $|S| \geq 2n$ . This observation leads to the following algorithm for MMV, which forms the basis for Kimbrel and Sinha’s algorithm [19]. Sample  $\alpha \sim \{1, \dots, 2n\}$ , and output “YES” if and only if  $AB\boldsymbol{\alpha} = \mathbf{0}$  for  $\boldsymbol{\alpha} := (1, \alpha, \alpha^2, \dots, \alpha^{n-1})^T$ . Using associativity, it is possible to compute this product as  $A(B\boldsymbol{\alpha})$  using  $O(n^2)$  arithmetic operations.

However, there is an issue with this algorithm: it requires computing powers of  $\alpha$  up to  $\alpha^{n-1}$ . These powers require  $\Omega(n)$  bits to represent for any integer  $\alpha \geq 2$ , and so performing arithmetic operations with them takes  $\Omega(n)$  time. To solve this, Kimbrel and Sinha instead consider the “test vector”  $\boldsymbol{\alpha}$  modulo an (arbitrary) prime  $2n \leq q \leq 4n$ , which they can find deterministically in  $O(n^2)$  time. They show that their algorithm is still correct with good probability (over the choice of  $\alpha$ ) with this modification.

Korec and Wiedermann [20] showed how to *deterministically* find a good  $\alpha$  for the above test – that is, a value  $\alpha$  such that  $p_i(\alpha) \neq 0$  if  $p_i$  is not identically zero – using *Cauchy’s bound*, which gives an upper bound on the magnitude of the largest root of a polynomial as a function of the polynomial’s coefficients. Namely, they just choose  $\alpha$  larger than Cauchy’s bound. (They note that the maximum magnitude of an entry in  $AB$  – and hence of a coefficient in

<sup>13</sup>A more general version of this theorem, which gives an algorithm whose running time depends both on the sparsity of the input matrices  $A, B$  and of their product  $AB$ , appears as [1, Theorem 1.7].

<sup>14</sup>To see this, note that in the latter case some row  $\mathbf{s}^T$  of  $AB$  must be non-zero, and let  $j^*$  be the index of the last non-zero entry in  $\mathbf{s}$ . Then for uniformly random  $\mathbf{x} \sim \{0, 1\}^n$ ,  $\Pr[AB\mathbf{x} = \mathbf{0}] \leq \Pr[(\mathbf{s}, \mathbf{x}) = 0] = \Pr[s_{j^*}x_{j^*} = -\sum_{k=1}^{j^*-1} s_k x_k] \leq 1/2$ . Moreover, this holds for matrices  $A, B$  over any ring  $R$ , and so Freivalds’s algorithm works for MMV over any ring  $R$ .

any of the polynomials  $p_i(x)$  – is at most  $n\mu^2$ , where  $\mu$  is the maximum magnitude of an entry in  $A$  or  $B$ .) Their algorithm uses only  $O(n^2)$  arithmetic operations, but again requires computing powers of  $\alpha$  up to  $\alpha^{n-1}$ , and therefore the algorithm has bit complexity  $\Omega(n^3)$ .

Additionally, we mention the work of Künnemann [21], which works for MMV over finite fields  $\mathbb{F}_q$  with  $q > n^2$  (he reduces MMV over the integers to MMV over such fields). His algorithm works by considering the bivariate polynomial  $f(x, y) = f_{A,B}(x, y) := \mathbf{x}^T A B \mathbf{y}$  for  $\mathbf{x} = (1, x, x^2, \dots, x^{n-1})$ ,  $\mathbf{y} = (1, y, y^2, \dots, y^{n-1})$ , where  $x$  and  $y$  are indeterminates, and the corresponding univariate polynomial  $g(x) = g_{A,B}(x) := f(x, x^n)$ . The coefficient of  $x^{(i-1)+(j-1)n}$  in  $g(x)$  (and of  $x^{i-1}y^{j-1}$  in  $f(x, y)$ ) is equal to  $(AB)_{i,j}$ , and so to decide whether  $AB = 0$  it suffices to decide whether  $g(x)$  (or  $f(x, y)$ ) is identically zero as a formal polynomial.<sup>15</sup> He shows that to do this it in turn suffices to decide whether  $g(\alpha^i) = 0$  for all  $i \in \{0, \dots, t-1\}$ , where  $\alpha \in \mathbb{F}_q$  is an element of order at least  $n^2$  and  $t = n^\delta$  is an upper bound on the sparsity of  $AB$ . Indeed, he notes that the system of equations  $g(1) = \dots = g(\alpha^{t-1}) = 0$  is a Vandermonde system of homogeneous linear equations in the at most  $t$  non-zero entries  $(AB)_{i,j}$  in  $AB$ , and so its only solution is the solution  $(AB)_{i,j} = 0$  for all  $1 \leq i, j \leq n$  (i.e., it must be the case that  $AB = 0$ ). To evaluate  $g$  on the  $t$  values  $1, \alpha, \dots, \alpha^{t-1}$  quickly, he uses a known result about fast multipoint polynomial evaluation.

We also note that MMV and its variants have also been studied from angles other than derandomization of Freivalds’s algorithm. Notably, [8] gave a  $O(n^{5/3})$ -time *quantum* algorithm for MMV, [15] studied the *Boolean* Matrix Multiplication Verification problem, and [13, 26] study the problem of *correcting* matrix products. I.e., they study the problem of *computing*  $AB$  given matrices  $A$ ,  $B$ , and  $C$  where  $\|AB - C\|_0$  is guaranteed to be small, which Künnemann showed is equivalent to OSMM.

Finally, we remark that other recent works including [9, 5, 2, 6] have also studied “barriers to SETH hardness”.

### 1.3 Open Questions

Of course, the main question that we leave open is whether Freivalds’s algorithm can be fully derandomized, i.e., whether there is a deterministic  $\tilde{O}(n^2)$ -time algorithm for MMV on  $n \times n$  matrices over finite fields  $\mathbb{F}_q$  with  $q \leq \text{poly}(n)$  and integer matrices with entries  $[-n^c, n^c]$  for constant  $c > 0$ . Additionally, it would be interesting to extend our results for MMV in the sparse regime to Output Sensitive Matrix Multiplication. The coding-theoretic techniques that we use seem amenable to this.

---

#### References

- 1 Amir Abboud, Karl Bringmann, Nick Fischer, and Marvin Künnemann. The time complexity of fully sparse matrix multiplication. In *SODA*, 2024.
- 2 Divesh Aggarwal, Huck Bennett, Zvika Brakerski, Alexander Golovnev, Rajendra Kumar, Zeyong Li, Spencer Peters, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan. Lattice problems beyond polynomial time. In *STOC*, 2023.
- 3 Joah Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *SODA*, 2021.
- 4 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost  $k$ -wise independent random variables. In *FOCS*, 1990.

---

<sup>15</sup> Indeed, [21] notes that this mapping from  $A, B$  to  $g(x)$  is a reduction from the All Zeroes Problem to Univariate Polynomial Identity Testing (UPIT).

- 5 Tatiana Belova, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, and Denil Sharipov. Polynomial formulations as a barrier for reduction-based hardness proofs. In *SODA*, 2023.
- 6 Tatiana Belova, Alexander S. Kulikov, Ivan Mihajlin, Olga Ratseeva, Grigory Reznikov, and Denil Sharipov. Computations with polynomial evaluation oracle: ruling out superlinear SETH-based lower bounds. *arXiv*, 2023. [arXiv:2307.11444](https://arxiv.org/abs/2307.11444).
- 7 Huck Bennett, Karthik Gajulapalli, Alexander Golovnev, and Evelyn S Warton. Matrix multiplication verification using coding theory. *arXiv preprint*, 2023. [arXiv:2309.16176](https://arxiv.org/abs/2309.16176).
- 8 Harry Buhrman and Robert Spalek. Quantum verification of matrix products. *arXiv*, 2004. [arXiv:quant-ph/0409035](https://arxiv.org/abs/quant-ph/0409035).
- 9 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *ITCS*, 2016.
- 10 Henry Cohn, Robert Kleinberg, Balázs Szegedy, and Christopher Umans. Group-theoretic algorithms for matrix multiplication. In *FOCS*, 2005.
- 11 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *FOCS*, 2023.
- 12 Rūsiņš Freivalds. Fast probabilistic algorithms. In *MFCS*, 1979.
- 13 Leszek Gasieniec, Christos Levcopoulos, Andrzej Lingas, Rasmus Pagh, and Takeshi Tokuyama. Efficiently correcting matrix products. *Algorithmica*, 79(2):428–443, 2017.
- 14 Jonathan I. Hall. Notes on coding theory. Available at <https://users.math.msu.edu/users/halljo/classes/codenotes/GRS.pdf>.
- 15 Wing-Kai Hon, Meng-Tsung Tsai, and Hung-Lung Wang. Verifying the product of generalized boolean matrix multiplication and its applications to detect small subgraphs. In *WADS*, 2023.
- 16 Mark A. Iwen, 2023. Personal Communication.
- 17 Mark A. Iwen and Craig V. Spencer. A note on compressed sensing and the complexity of matrix multiplication. *Information Processing Letters*, 109(10):468–471, 2009. [doi:10.1016/j.ipl.2009.01.010](https://doi.org/10.1016/j.ipl.2009.01.010).
- 18 Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *ICALP*, 2015.
- 19 Tracy Kimbrel and Rakesh Kumar Sinha. A probabilistic algorithm for verifying matrix products using  $O(n^2)$  time and  $\log_2 n + O(1)$  random bits. *Information Processing Letters*, 45(2):107–110, 1993.
- 20 Ivan Korec and Jiří Wiedermann. Deterministic verification of integer matrix multiplication in quadratic time. In *SOFSEM*, 2014.
- 21 Marvin Künnemann. On nondeterministic derandomization of Freivalds’ algorithm: Consequences, avenues and algorithmic progress. In *ESA*, 2018.
- 22 François Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *SODA*, 2024.
- 23 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *SODA*, 2018.
- 24 Ron M. Roth and Abraham Lempel. On MDS codes via Cauchy matrices. *IEEE Transactions on Information Theory*, 35(6):1314–1319, 1989.
- 25 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *SODA*, 2024.
- 26 Yu-Lun Wu and Hung-Lung Wang. Correcting matrix products over the ring of integers. *arxiv*, 2023. [arXiv:2307.12513](https://arxiv.org/abs/2307.12513).
- 27 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002.



# Interactive Coding with Unbounded Noise

Eden Fargion ✉ 

Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Ran Gelles ✉ 

Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel

Meghal Gupta ✉ 

University of California, Berkeley, CA, USA

---

## Abstract

---

Interactive coding allows two parties to conduct a distributed computation despite noise corrupting a certain fraction of their communication. Dani et al. (Inf. and Comp., 2018) suggested a novel setting in which the amount of noise is unbounded and can significantly exceed the length of the (noise-free) computation. While no solution is possible in the worst case, under the restriction of *oblivious* noise, Dani et al. designed a coding scheme that succeeds with a polynomially small failure probability.

We revisit the question of conducting computations under this harsh type of noise and devise a computationally-efficient coding scheme that guarantees the success of the computation, except with an *exponentially* small probability. This higher degree of correctness matches the case of coding schemes with a bounded fraction of noise.

Our simulation of an  $N$ -bit noise-free computation in the presence of  $T$  corruptions, communicates an optimal number of  $O(N + T)$  bits and succeeds with probability  $1 - 2^{-\Omega(N)}$ . We design this coding scheme by introducing an intermediary noise model, where an oblivious adversary can choose the locations of corruptions in a worst-case manner, but the effect of each corruption is random: the noise either flips the transmission with some probability or otherwise erases it. This randomized abstraction turns out to be instrumental in achieving an optimal coding scheme.

**2012 ACM Subject Classification** Theory of computation → Interactive computation; Mathematics of computing → Coding theory; Computing methodologies → Distributed algorithms

**Keywords and phrases** Distributed Computation with Noisy Links, Interactive Coding, Noise Resilience, Unbounded Noise, Random Erasure-Flip Noise

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.43

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.09463> [10]

**Funding** *Ran Gelles*: partially supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, Grant No. 2020277.

**Acknowledgements** E. Fargion and R. Gelles would like to thank Paderborn University for hosting them while part of this research was done. R. Gelles would like to thank CISPA – Helmholtz Center for Information Security for hosting him.

## 1 Introduction

In many distributed systems nowadays, communication channels suffer various types of noise and interference that may corrupt information exchanged between devices. *Interactive coding*, initiated by the seminal work of Schulman [25, 26] (see also [12]), allows two or more devices to correctly complete their computation despite channel noise, by adding only a moderate amount of redundancy to the computation. The capability of an interactive coding scheme usually depends on the specific type of noise it is designed to withstand. For instance, when



© Eden Fargion, Ran Gelles, and Meghal Gupta;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 43; pp. 43:1–43:16



Leibniz International Proceedings in Informatics

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

the noise can flip bits, interactive coding schemes withstand up to a fraction of  $1/6$  of flipped bits [8, 20] (a fraction of  $1/4$  can be withstood over channels with larger alphabets [5]); when the noise erases bits (i.e., replaces a bit with a special erasure mark  $\perp$ ), then a fraction of  $1/2$  of bit erasures can be withstood [11, 20], which also applies for larger alphabets [8]. When messages can be inserted and deleted, the maximal corruption rate is again  $1/4$ , see [4, 27].

In a recent work, Dani, Hayes, Movahedi, Saia, and Young [7] suggested a different and interesting model for interactive coding in which the amount of noise is *unbounded*. That is, the number  $T$  of corruptions that affects a given execution, can be arbitrary. Note that this number  $T$  is unknown to the coding scheme; this is in contrast to the standard model of interactive coding, where a limit on the fraction of corrupted transmissions is known by all devices. The scheme in [7] correctly computes any two-party computation that takes  $N$  rounds without noise, by communicating  $N + O(T + \sqrt{N(T+1)\log T})$  bits and succeeds with probability  $1 - O(1/N \log N)$ .

In a nutshell, the idea of the scheme in [7] is as follows. Every message sent between the parties contains the round number it corresponds to and a signature. A device checks that the signature is valid before processing a received message. If the signature does not check out, the device ignores that communication. The coding scheme tracks the progress of both parties via the added information of the round number, so that corrupted messages are re-transmitted until they arrive correctly at the other side.

One significant drawback of the above approach, is that the noise might corrupt a message along with its signature so that the receiver believes that the signature is correct. This occurs with exponentially small probability in the length of the signature, which leads to the polynomially-small failure probability of the scheme. In other words, the scheme in [7] assumes that the noise *never creates a valid signature* and settles with a failure probability of magnitude  $1/N \log N$ .

In this work we aim to achieve an interactive coding scheme that can withstand an unbounded amount of noise, yet, with failure probability exponentially small in  $N$ , similar to most previous work on interactive coding (e.g., [26, 18, 2, 19]). This effectively means that the coding scheme must cope with corrupted messages being processed by some device. That is, the coding scheme must be resilient to the event, that occurs with polynomially small probability in  $N$ , where both the message and the signature are corrupted in a matching way.

Our main result is a coding scheme that is resilient to an arbitrary and a priori unknown number  $T$  of bit flips, with exponentially small failure probability.

► **Theorem 1.1 (Main).** *Given any two-party binary interactive protocol  $\pi$  of length  $N$ , there exists an efficient randomized protocol  $\Pi$  of length  $O(N + T)$  that simulates  $\pi$  with probability  $1 - 2^{-\Omega(N)}$  over a binary channel in the presence of an arbitrary and a priori unknown number  $T$  of corruptions. The noise is assumed to be independent of the parties' randomness.*

We note that the scheme assumes *oblivious noise* in the sense that the  $T$  corrupted transmissions are selected at the beginning of the computation (as a function of the coding scheme and the parties' inputs) and is independent of the parties' (private) randomness. This assumption is crucial, as no coding scheme withstands an unbounded amount of noise that is non-oblivious [7, Theorem 6.1].

## 1.1 Techniques

**Towards an optimal scheme: the code concatenation approach.** The immediate approach towards an improved coding scheme for an unbounded amount of corruptions is of *code concatenation*, namely composing two layers of interactive code. The inner layer would be



responsible for transmitting bits over the channel despite the unbounded amount of noise (e.g., [7, 15]). The outer layer would then “see” only a limited amount of noise (which passes the inner layer with polynomially-small probability) and perform a standard interactive coding (e.g., [25, 26, 5, 21, 19, 14, 20]) using these bits.

Unfortunately, such an approach faces severe difficulties. For the inner layer, the scheme of Dani et al. [7] assumes that no corrupted message is accepted by a party. That is, a message can either be correct or marked as invalid. Requiring the parties to process incorrect messages might cause their inner state to differ in a way that could not be recovered by the scheme. That is, a single corrupted message (that is believed to be correct by one of the parties) might cause the parties to “lose sync”, so that the parties do not agree anymore on when the next phase of the scheme begins and ends, or whether the scheme has terminated or not. The scheme will not recover from this fault because the synchronization information would be sent by one party at certain rounds but expected by the other party at different rounds. The other option for the inner layer is the scheme by Gelles and Iyer [15] designed to withstand an unbounded amount of *erasures*, and thus, on its surface, does not fit our purpose.

**The randomized erasure–flip model.** The key towards solving the above conundrum stems from defining a new random noise model that we name the *unbounded probabilistic Erase–Flip noise model* (UPEF). This model (formally defined in Section 2.2) still allows an unbounded number of corruptions determined by the adversary in an oblivious way. However, when the  $i$ -th transmission is corrupted by the adversary, the *effect* of the corruption is random: the transmitted bit is flipped with some probability  $p_i$  or erased with the complementary probability. The probabilities  $\{p_i\}_{i \in \mathbb{N}}$  are parameters of the model and can be determined by the algorithm’s designer. In a sense, this type of noise matches the effect oblivious noise has on messages that are protected with a signature: with some probability the corruption is detected (the signature does not verify) and the message is marked as corrupted, i.e., erased. On the other hand, with some small probability the corruption is such that the signature verifies the corrupted message; in this case we have a flip. The probabilities are determined by the length of the signatures in use.<sup>1</sup>

This novel randomized model is much simpler to handle, and facilitates the design and analysis of optimal coding schemes. Furthermore, any scheme designed for this model can be translated into a coding scheme that works in the standard unbounded flips (UF) noise model, by employing signatures of respective length to match the erasure–flip probabilities of the UPEF model. Therefore, this model serves as a crucial tool for obtaining optimal coding schemes in the standard model.

Switching to the UPEF model allows us to use the scheme in [15] as the inner code of our concatenated coding scheme, in an almost as-is fashion: by smartly setting the probabilities  $\{p_i\}_{i \in \mathbb{N}}$ , we can guarantee, with very high probability, that any execution experiences an unbounded number of erasures but only a bounded number of bit-flips. The scheme in [15] withstands the erasures and delivers the non-erased bits (either correct or not) to the outer layer, which should be able to cope with this limited amount of bit flips.

One problem still remains, but to explain it, we must first explain how the [15] scheme works. In a nutshell, the parties simulate the underlying (noiseless) protocol  $\pi$  bit-by-bit, where the scheme adds to each bit the parity (mod 2) of the corresponding round number.

<sup>1</sup> We use AMD codes [6] to generate signatures, see the full version of this paper [10, Appendix A] for the exact details.

The scheme works in *challenge-response*-style iterations: The first party (Alice) begins by sending the bit of the next round of  $\pi$  along with the parity of that round (*the challenge*). Bob receives this bit, and if the parity corresponds to the round number he expects, he records this bit and replies with the next bit of  $\pi$  along with the parity of that round in  $\pi$  (*the response*). When this reply reaches Alice, and the parity is correct, Alice records the bit from Bob and moves on to the next iteration. In any case of erasures or if the parity mismatches, the receiver ignores the received message. The analysis in [15] shows that this single parity bit suffices to keep track of the progress despite an unbounded amount of erasures.

However, in the UPEF model, a bit flip can either corrupt the content bit (i.e., the next simulated round of  $\pi$ ) or the parity bit sent along! Corrupting the parity bit damages the correctness of the [15] scheme, but this is the only way the noise can affect correctness. Throughout a detailed case analysis, we prove that corrupting the parity bit has the sole effect of making the parties *out-of-sync*, in the sense that one party advances to the next round in  $\pi$ , while the other does not. Luckily, this type of out-of-sync corruption was already considered in the interactive-coding community, initiated by the work of Braverman, Gelles, Mao, and Ostrovsky [4], which presented a non-efficient scheme that withstands a noise level of up to  $1/18$  fraction of the rounds, where “noise” here means insertions and deletions producing out-of-sync events as described above. That work was followed by a work by Sherstov and Wu [27], who showed that a variant of the [4] scheme withstands the optimal level of noise, namely, up to  $1/4$  of the rounds, and by a work by Haeupler, Shahrasbi, and Vitercik [22], who presented an *efficient* scheme, based on synchronization strings, with noise resilience of  $1/44$ .

Therefore, we can set the scheme in [22] (denoted HSV hereinafter) as the outer layer in our construction, and set the probabilities  $\{p_i\}_{i \in \mathbb{N}}$  such that the total number of insertion and deletion errors will not surpass the threshold expected by the HSV scheme, except with an exponentially small probability. This construction achieves our goal of obtaining a coding scheme in the UPEF model, with optimal length of  $O(N + T)$ , and an exponentially small failure probability.

Unfortunately, once converting this optimal UPEF scheme back to the standard UF model, the overhead increases severely. In particular, the way we set the probabilities  $\{p_i\}_{i \in \mathbb{N}}$  implies a logarithmic overhead on the size of the signatures, leading to a sub-optimal scheme of length  $O((N + T) \log(N + T))$  in the UF model. To avoid this increase in communication, we must maintain the probabilities  $\{p_i\}_{i \in \mathbb{N}}$  “large”, and design a new scheme that is still optimal in the UPEF model despite the high values of  $\{p_i\}_{i \in \mathbb{N}}$ .

**Obtaining an optimal scheme: the iterative approach.** In order to obtain a UF-optimal scheme, we take a different approach, namely, we execute an increasing-length instances of a “standard” interactive coding [7]. As before, we start by constructing a coding scheme over the UPEF model. Our goal now is to maximize the  $p_i$ ’s as much as possible. The main idea is as follows. Let’s fix each  $p_i$  to some constant, say,  $2/3$ . Now, any corruption in the UPEF model will cause an erasure with a fixed probability of  $1/3$ . The number of erasures a party observes is a good estimate of the level of noise during the same transmissions. Hence, the parties can continue running the scheme again and again, until they believe the noise level was low enough to produce the correct output.

In more detail, Alice and Bob run an efficient interactive coding scheme resilient to a constant fraction of adversarial flips (e.g., [2, 19]). After executing the scheme, Alice and Bob count the number of erasures observed during the execution and estimate (with high probability) the fraction of corruption they experienced. They communicate this estimate

to each other, and decide how to continue accordingly. If the noise level seems sufficiently low, the resilient scheme must have produced the correct output, and the parties can safely terminate. Otherwise, the parties re-run the interactive scheme, doubling its length. They repeat this action until they reach an execution where the noise level is low enough to guarantee the success of the underlying interactive coding scheme.

With a correct choice of parameters, this results in a UPEF scheme of length  $O(N + T)$ . However, since all  $\{p_i\}_{i \in \mathbb{N}}$  are fixed to a constant, once we translate this scheme into a UF scheme, we keep its length up to a constant and obtain an optimal length of  $O(N + T)$  in this case as well.

We note that a communication complexity of  $\Theta(N + T)$  is tight for the UF model. A lower bound of  $\Omega(N + T)$  is immediate by considering the case where the adversary corrupts the entirety of the communication between Alice and Bob for  $\Theta(T)$  rounds, e.g., by flipping each bit with probability  $1/2$ , thereby not allowing any information to cross the channel during these rounds. After this corruption,  $N$  rounds are still needed to complete the protocol without noise.

## 1.2 Related Work

As mentioned above, the field of interactive coding was initiated by the work of Schulman [25, 26]. Following this work, many two-party interactive coding schemes were developed, with the goal to optimize various properties, such as efficiency, communication rate, and noise resilience [5, 18, 2, 23, 21, 13, 3, 14, 20]. Two-party coding schemes for different types of noise, such as erasures or insertions and deletions, appeared in [11, 13, 8, 4, 22, 27, 9, 20]. See [12] for an extensive survey on this field.

Closest to our work are coding schemes that withstand an unbounded amount of corruption. As mentioned above, Dani et al. [7] developed a randomized scheme that deals with an unbounded amount  $T$  of oblivious bit-flips, succeeds with high probability, and simulates any  $\pi$  of length  $N$  in  $N + O(T + \sqrt{N(T + 1) \log T})$  rounds. Gelles and Iyer [15] developed a deterministic scheme that deals with an unbounded amount  $T$  of (not necessarily oblivious) erasures in at most  $2N + 4T$  communication rounds. For the multiparty setting, Aggarwal, Dani, Hayes, and Saia [1] developed a coding scheme that correctly simulates any protocol over an arbitrary network withstanding an unbounded amount of oblivious corruptions in  $\tilde{O}(N + T)$  rounds, suppressing logarithmic terms.

## 1.3 Paper Outline

We set up the UPEF and UF models, recall the insertion-deletion model, and review interactive coding protocols in Section 2. In Section 3 we describe an optimal UPEF coding scheme that follows a code concatenation approach. Its analysis is deferred to the full version of this paper [10, Appendix A]. Finally, in Section 4 we describe an optimal coding scheme in the UPEF model that follows an iterative approach. We then show how to translate it into an optimal UF coding scheme.

## 2 Preliminaries

**Notations.** For an integer  $n \in \mathbb{N}$  we use  $[n] = \{1, 2, 3, \dots, n\}$ . All logarithms are taken to base 2 unless otherwise mentioned. For two strings  $a, b$  we denote by  $a \circ b$  their concatenation. We will use  $\bigcirc_{k=1,2,\dots,\ell} a_k \triangleq a_1 \circ a_2 \circ \dots \circ a_\ell$  to abbreviate the concatenation of multiple strings. We use  $O_\varepsilon(\cdot)$ ,  $\Theta_\varepsilon(\cdot)$ , etc., to explicitly remind that the constant inside the  $O(\cdot)$  may depend on (the constant)  $\varepsilon$ .

## 2.1 Interactive Protocols and Coding Schemes

Consider two parties, Alice and Bob, having inputs  $x, y \in \{0, 1\}^k$  respectively, who wish to compute some function  $f(x, y)$  by communicating over a channel with alphabet  $\Sigma$ . Towards that goal, Alice and Bob use an *interactive protocol* composed of two algorithms  $\pi = (\pi_a, \pi_b)$  for Alice and Bob, respectively. These algorithms assume a common clock known by both parties (i.e., the protocol is synchronized) and determine, for each party in each round (timestep), whether the party (1) has to send a message in that round, (2) which symbol the party sends, and (3) if the party should terminate in that round and which output should it give.

Each party records all the messages it receives during the execution of the protocol. The collection of these records is the party's *transcript*. We assume that  $\pi$  has *fixed order of speaking*; this means that in each round exactly one party is transmitting a symbol (the other party listens), and the identity of the transmitting party in a given round is predetermined and independent of the parties' inputs. In particular, a protocol in which Alice speaks in odd rounds, and Bob speaks in even rounds is said to be of an *alternating order*. Note that if  $\pi$  is not alternating, then it can be converted to an alternating-order protocol while increasing the communication complexity by a factor of at most 2. We say that a protocol is *k-alternating*, for some  $k \in \mathbb{N}$ , if during its execution each party transmits bulks of  $k$  bits. The *length* of a protocol is defined to be the number of rounds it includes until both parties have terminated.

**Noisy channels and coding schemes.** Now, assume that the parties are connected by a *noisy channel*. Formally, given an input and output alphabets  $\Sigma_{in}, \Sigma_{out}$ , respectively, a single utilization of a noisy channel is the (possibly randomized) function  $C : \Sigma_{in} \rightarrow \Sigma_{out}$ .

We can now discuss protocols that perform over noisy channels. We say that a protocol  $\pi'$  *simulates*  $\pi$  over the noisy channel  $C$ , if for any inputs  $(x, y)$ , after executing  $\pi'$  over the noisy channel  $C$ , the parties can output their transcripts in an execution of  $\pi$  over a noiseless channel with inputs  $(x, y)$ . When the channel noise or the algorithm  $\pi'$  are probabilistic, we say that  $\pi'$  *simulates*  $\pi$  *with probability*  $p$  if the probability that the parties' output equal the transcript of  $\pi$  is at least  $p$ , for any inputs pair.

A *coding scheme* (for some given noisy channel  $C$ ) is a function  $CS$ , whose input is a noiseless protocol  $\pi$ , and its output is a protocol  $\pi' = CS(\pi)$  which simulates  $\pi$  over the channel  $C$ . When the channel noise or the scheme are probabilistic, we say that the coding scheme has success probability  $p$ , if for any  $\pi$ , the protocol  $\pi' = CS(\pi)$  simulates  $\pi$  with probability  $p$ .

## 2.2 Noise Models

As alluded to in the introduction, our scheme is designed to withstand an unbounded amount of (oblivious) bit flips. However, we design the scheme by reducing the unbounded-flip model to a different noise model with unbounded probabilistic erasures and flips. Furthermore, the effect of probabilistic erasures and flips noise on the inner layer of our coding scheme is such that the outer layer "sees" insertion and deletion noise. We will now define these three noise models in turn.

**The Unbounded Flip noise model (UF).** Our main noise model is the unbounded flip noise model, set forth by Dani et al. [7]. Given a specific execution of  $\pi'$  with inputs  $(x, y)$ , the adversary sets a *noise corruption pattern*  $E \subset \mathbb{N}$  such that the amount of noise,  $|E|$ , satisfies  $|E| = T$  for some number  $T \in \mathbb{N}$  decided by the adversary. The noise pattern can

be set as a function of  $\pi, x, y$  but is independent of any randomness the parties might have (i.e., an *oblivious* noise). The noise pattern  $E$  determines which bits get flipped during the execution of  $\pi$ . Namely, if  $i \in E$ , then the  $i$ -th transmitted bit in  $\pi$  will be flipped. Otherwise, the bit goes through uncorrupted. Note that  $T$  might be arbitrary. When one of the parties terminates, the channel sends zeros to the another party, which may be flipped by the adversary.

**The Unbounded Probabilistic Erasure-Flip noise model (UPEF).** Our coding scheme in this work is designed and analyzed within the following noise model, that combines both erasure and flip noise. This model naturally appears when executing a protocol in the UF model while each message contains a (probabilistic) signature or a message authentication tag that indicates its validity.

In this model, the parties are connected via a noisy communication channel  $C : \{0, 1\} \rightarrow \{0, 1, \perp\}$ , which can either flip bits or erase them (denoted by the erasure mark  $\perp$ ). Similar to the UF model, given any specific execution of  $\pi'$  with inputs  $(x, y)$ , the rounds which are corrupted are predetermined by an adversary that knows  $\pi', x, y$  and the inputs but not the parties' private randomness. This corruption is described via the noise pattern  $E \subset \mathbb{N}$ , where  $i \in \mathbb{N}$  means that the  $i$ -th round is corrupted; otherwise, the bit arrives at the other side intact. When a round is corrupted, the effect is as follows: the bit is flipped with some probability  $p_i$  or is erased with probability  $1 - p_i$ . The probabilities  $\{p_i\}_{i \in \mathbb{N}}$  are parameters of the model and will be specified later.

Terminating in the UPEF is different from terminating in the UF. When Alice terminates, the channel transmits a special “silence” symbol, namely, “ $\square$ ”. Upon the reception of this special symbol, Bob knows that Alice has quit, and terminates as well.

Similar to the UF model, we restrict the discussion to noise patterns in which the total number of corrupted rounds is finite. That is, there exists some number  $T \in \mathbb{N}$ , unknown to the parties and  $\pi$ , such that  $|E| = T$ .

**The Insertion-Deletion noise model.** The insertion-deletion noise model [4], which we briefly describe here, is important for our analysis of the concatenated coding scheme.

In this model we consider *alternating* interactive protocols  $\pi'$ , where no common clock is assumed by the parties. Instead, Alice sends the first symbol (round 1), and Bob is idle until receiving this symbol. Once the first symbol is obtained by Bob he transmits a symbol back to Alice (round 2). Alice will execute round 3 once receiving this symbol, and so on. The noise is allowed to either corrupt a symbol (i.e., the receiver will obtain a different symbol from the one sent, a *substitution*), or to completely delete the symbol, so that the receiver receives nothing. In the latter case, the protocol is “stuck” as both parties await an incoming symbol to proceed. To avoid getting stuck, the noise must inject a new symbol towards the sender of the symbol that got deleted. This causes the parties to get *out of sync*, that is, one of them will believe the current round is  $i$ , while the other will believe the current round is  $i + 2$ . See also [4, 27, 22, 9, 16, 17].

In [22], the authors give an efficient constant-rate coding scheme for insertions and deletion noise, which we will use in our construction.

► **Theorem 2.1** (Theorem 1.2 in [22]). *For any alternating protocol  $\pi$  of length  $n$  and for any  $\varepsilon > 0$ , there exists an efficient randomized protocol  $\pi'$  simulating  $\pi$  in presence of  $\delta = 1/44 - \varepsilon$  fraction of edit-corruptions, whose length is  $\Theta_\varepsilon(n)$  and succeeds with probability  $1 - 2^{-\Theta(n)}$ . The alphabet size of  $\pi'$  is  $\Theta_\varepsilon(1)$ .*

We assume that at its termination,  $\pi'$  has an output, which equals to the output of  $\pi$  (under the conditions in the theorem).

### 3 A UPEF-optimal coding scheme via code concatenation

In this section we give an optimal UPEF coding scheme, based on code-concatenation approach (Algorithms 1 and 2). The analysis of the scheme, presented in the full version of this paper [10, Appendix A], proves the following Theorem.

► **Theorem 3.1.** *Given any two-party binary interactive protocol  $\pi$  of length  $N$ , there exists some constant  $C$  and an efficient randomized protocol  $\Pi$  of length  $O(N + T)$  that simulates  $\pi$  with probability  $1 - 2^{-\Omega(N)}$  over a binary channel in the presence of an arbitrary and a priori unknown number  $T$  of probabilistic Erasure–Flip corruptions, with  $p_i = \min \left\{ \frac{CN}{i^2}, \frac{1}{2} \right\}$ .*

Let  $\pi$  be a binary alternating protocol that assumes noiseless channels of length  $|\pi| = N$ . Our goal is to simulate  $\pi$  in the UPEF model. Let  $\pi'$  be the randomized protocol obtained from  $\pi$  via Theorem 2.1, by setting  $\delta = 1/45$  (i.e.,  $\varepsilon = 1/1980$ ). We denote  $|\pi'| = N'$ . Denote the alphabet of  $\pi'$  by  $\Sigma'$  and note that its size is constant,  $|\Sigma'| = \Theta(1)$ .

Our coding scheme (Algorithms 1 and 2) simulates the communication of  $\pi'$ , symbol by symbol. As the channel in the unbounded probabilistic Erase–Flip noise model is binary, the parties communicate the binary representations of the symbols in  $\pi'$ . Therefore, during each iteration of the simulation, Alice sends a symbol to Bob using  $\log |\Sigma'|$  bit transmissions, and expects a symbol reply from Bob.

■ **Algorithm 1** Simulation over Erasure and Substitution Channel with Unbounded Noise (Alice).

---

**Input:** An alternating binary protocol  $\pi$  of length  $N$ , an input  $x$   
**Initialize:** Let  $\pi'$  be the protocol simulating  $\pi$  given by Theorem 2.1, setting  $\varepsilon = 1/1980$ . Let  $N' = |\pi'|$  and assume  $\Sigma'$  (the alphabet of  $\pi'$ ) is a power of two.

---

```

A.1  $\mathcal{T}_a \leftarrow \emptyset, r_a \leftarrow 0$ 
A.2 while  $r_a < \frac{N'}{2}$  do
A.3   // Send Message
A.4    $r_a \leftarrow r_a + 1$ 
A.5    $m_{send} \leftarrow \pi'(x \mid \mathcal{T}_a)$ 
A.6    $\mathcal{T}_a \leftarrow \mathcal{T}_a \circ m_{send}$ 
A.7   send  $(m_{send}, r_a \bmod 2)$  ▷  $k$  transmissions
A.8
A.9   // Receive Message
A.10  receive  $m' = (m_{rec}, r_{rec})$  ▷  $k$  transmissions
A.11  if  $m'$  does not contain  $\perp$  and  $r_{rec} = r_a \bmod 2$  then
A.12     $\mathcal{T}_a \leftarrow \mathcal{T}_a \circ m_{rec}$ 
A.13  else
A.14    delete the last symbol of  $\mathcal{T}_a$ 
A.15     $r_a \leftarrow r_a - 1$ 
A.16  end if
A.17 end while
A.18 Output the output given by  $\pi'$ 

```

---

The simulation of  $\pi'$  employs a challenge-response paradigm, where Alice sends a symbol (the challenge) and expects one back (the response). The parties maintain a counter to track their respective progress, namely, the variables  $r_a$  and  $r_b$ , which represent the number of successful iterations observed by Alice and Bob, respectively. Every time Alice and Bob send a symbol, they attach to it the parity (mod 2) of their own counter. Hence, the simulation is  $k$ -alternating, with  $k = \lceil \log |\Sigma'| \rceil + 1$ .

---

**Algorithm 2** Simulation over Erasure and Substitution Channel with Unbounded Noise (Bob).
 

---

**Input:** An alternating binary protocol  $\pi$  of length  $N$ , an input  $x$

**Initialize:** Let  $\pi'$  be the protocol simulating  $\pi$  given by Theorem 2.1, setting  $\varepsilon = 1/1980$ . Let  $N' = |\pi'|$  and assume  $\Sigma'$  (the alphabet of  $\pi'$ ) is a power of two.

```

B.1  $\mathcal{T}_b \leftarrow \emptyset, r_b \leftarrow 0, err \leftarrow 0, m \leftarrow (0, 0)$ 
B.2 while  $m' \neq \square$  do
B.3   // Receive Message
B.4   receive  $m' = (m_{rec}, r_{rec})$  ▷  $k$  transmissions
B.5   if  $m'$  does not contain  $\perp$  and  $r_{rec} \neq r_b \pmod 2$  then
B.6      $\mathcal{T}_b \leftarrow \mathcal{T}_b \circ m_{rec}$ 
B.7      $err \leftarrow 0$ 
B.8   else
B.9      $err \leftarrow 1$ 
B.10  end if
B.11
B.12  // Send Message
B.13  if  $err = 0$  then
B.14     $r_b \leftarrow r_b + 1$ 
B.15     $m_{send} \leftarrow \pi'(y \mid \mathcal{T}_b)$ 
B.16     $\mathcal{T}_b \leftarrow \mathcal{T}_b \circ m_{send}$ 
B.17    send  $m \leftarrow (m_{send}, r_b \pmod 2)$  ▷  $k$  transmissions
B.18  else
B.19    send  $m$  ▷  $k$  transmissions;  $m$  from memory
B.20  end if
B.21 end while
B.22 Output the output given by  $\pi'$ 

```

---

When Alice receives a symbol (as a response to the challenge she has previously sent), she checks the counter value attached to it: if it matches her expected counter parity (mod 2), she “believes” this challenge-response iteration, delivers the received symbol to  $\pi'$  and increases  $r_a$  by 1; otherwise, she ignores the reply and tries again in the next iteration.

Bob acts in an analogous manner: if the information received from Alice matches the counter parity (mod 2) he is expecting, then he “believes” the received symbol, delivers it to  $\pi'$ , increases  $r_b$  by 1, obtains from  $\pi'$  the next symbol to communicate to Alice, and sends Alice this symbol and the parity of  $r_b$ . If the information from Alice does not match Bob’s expectation, he ignores this transmission and replies with the previous symbol computed by  $\pi'$  (along with the parity that corresponds to that symbol). When a party “believes” an iteration, it appends the received and transmitted symbols of the iteration to its transcript,  $\mathcal{T}_a$  or  $\mathcal{T}_b$ , respectively. This transcript records all the symbols communicated so far (by  $\pi'$ ) during the “successful” iterations of Algorithms 1 and 2.

To summarize, in each iteration of the loop, Alice generates the next message of  $\pi'$ , denoted  $m \in \Sigma'$ , based on her current transcript  $\mathcal{T}_a$  and her input  $x$ , i.e.,  $m = \pi'(x \mid \mathcal{T}_a)$ . Alice (temporarily) adds  $m$  to  $\mathcal{T}_a$ , and sends its binary representation to Bob, along with the parity of  $r_a$ . After receiving a  $k$ -bit message  $(m_{rec}, r_{rec})$  from Alice, Bob checks that none of the  $k$  bits have been erased (denoted by  $\perp$ ) and that  $r_{rec}$  is opposite to his parity (since Alice added a new symbol and he did not, yet). If everything matches, Bob adds  $m_{rec}$  to his transcript  $\mathcal{T}_b$ , increases  $r_b$  by 1, computes the next message  $m' = \pi'(y \mid \mathcal{T}_b)$  and the new parity of  $r_b$ , and transmits them to Alice. On the other hand, if Bob notices any erasures or the mismatch of the parity, he ignores Alice’s new symbol and replies with the latest computed  $(m', r_b)$  recorded in his memory.

At the end of the iteration, Alice receives a message and a parity from Bob; if there were no erasures and the received parity matches  $r_a$ , she adds that message to her transcript. Otherwise, Alice deletes the (temporary) message she added at the beginning of this iteration.

The algorithm ends once the length of  $\mathcal{T}_a$  reaches the length of  $\pi'$  (for Alice) or when a special symbol  $\square$ , sent by the channel when Alice quits, is received by Bob. We discuss termination in this model and the implication of assuming this special symbol in the full version of this paper [10, Appendix B.1].

The complete detailed analysis of the coding scheme (Algorithms 1 and 2) and the proof of Theorem 3.1, are deferred to the full version of this paper [10, Appendix A].

## 4 A UF Scheme with Optimal Communication

The concatenated scheme in Section 3, while optimal in the UPEF model, implies a UF scheme of length  $O((N + T) \log(N + T))$  in which the  $i$ -th bit is replaced with an AMD code [6] of length  $O(\log(p_i^{-1}))$ ; see the full version of this paper [10, Appendix B] for details. In this section, we take a different approach towards constructing an optimal coding scheme in the UPEF model, namely by executing increasing-length coding schemes in an iterative fashion. This approach eventually leads to an optimal-communication UF scheme.

Here, Alice and Bob utilize a “standard” interactive coding scheme for substitutions and estimate the experienced level of noise. If the estimated noise level is too high, Alice and Bob repeat the execution with a larger amount of redundancy. When the noise level is low enough, the interactive scheme guarantees the success of the computation, and the parties can terminate. This approach, in addition to leading to a UF scheme with optimal communication, is also much simpler and easier to analyze than the scheme of Section 3. The key difference is that, all we need in order to estimate the noise level well, is that the probabilities  $\{p_i\}_{i \in \mathbb{N}}$  are bounded below by a constant, rather than converging to 0. This aligns perfectly with obtaining optimal complexity, as smaller  $\{p_i\}_{i \in \mathbb{N}}$  imply longer encodings.

The scheme in this section utilizes a slight variant of the UPEF model, denoted the modified UPEF (mUPEF) model, which we now describe. Let a noise pattern  $E \subset \mathbb{N}$  be determined adversarially. As before, if  $i \notin E$ , the  $i$ -th transmitted bit reaches the other party intact. For  $i \in E$ , the  $i$ -th transmitted bit is still erased with probability  $1 - p_i$ , and corrupted with probability  $p_i$ . However, the corruption here is not necessarily a bit flip as in the original UPEF. Instead, the adversary determines whether the bit is flipped, erased, or not corrupted at all. The probabilities  $\{p_i\}_{i \in \mathbb{N}}$  are parameters of the model. However, we will actually set them all to have the same value. That is,  $\forall i, p_i = 2/3$ .

In our scheme, we set  $p_e = 1 - p_i = 1/3$  as a lower bound on the probability that a bit is erased. Similar to the UF model, we will assume that when Alice terminates, the channel implicitly sends Bob a default symbol (e.g., a zero).

In the following sections we describe and analyze a coding scheme in the mUPEF model, with optimal  $O(N + T)$  communication that, due to our choice of  $p_e = 2/3$ , results with a UF scheme with  $O(N + T)$  communication as well.

**An optimal mUPEF Coding Scheme.** For the underlying substitution-resilient interactive coding scheme, we can take any (efficient) 2-party scheme with binary alphabet that is resilient to a constant fraction of adversarial noise, e.g., [2, 19]. In particular, let us assume an interactive coding scheme that simulates any (noiseless) protocol  $\pi$  in the presence of up to 0.1 adversarial substitutions with a constant rate over the binary alphabet. Denote the substitution-resilient version by  $\pi'$ , so  $|\pi'| = O(|\pi|) = O(N)$ . We assume that  $\pi'$  is



alternating. We additionally assume that the communication of  $\pi'$  includes a constant fraction of ones. To be concrete, out of the  $|\pi'|/2$  bits Alice sends, at least  $|\pi'|/8$  are the bit 1, in any execution of  $\pi'$ . We must have this property, because in our scheme, a long sequence of zeros will indicate termination. For that reason, we want that  $\pi'$  will not send a long sequence of zeros. This can be achieved, for instance by making the parties communicate a 1 every alternate round, or by means of randomization (see, e.g., [13]).

We proceed to describe our mUPEF resilient scheme  $\Pi$  simulating the substitution-resilient  $\pi'$  defined above. The execution of  $\Pi$  consists of iterations, where the  $i$ -th iteration,  $i = 0, 1, 2, \dots$ , takes  $2L_i$  rounds with  $L_i = |\pi'|2^i$ . The  $i$ -th iteration can be broken down into two parts, each of length  $L_i$ . In the first part, the parties execute  $\pi'$  from scratch, padded to length  $L_i$ ; in this padded protocol, each bit of  $\pi'$  is sent  $2^i$  times, and decoding is performed by majority (defaulting to 0 on ties, considering erased copies as zeros). In the second part of iteration  $i$ , only Bob speaks. He sends Alice the *success string*  $0^{L_i}$  if and only if he observed less than  $0.001p_e L_i$  erasures in the first part; otherwise Bob sends the *error string*  $1^{L_i}$ .

Alice terminates at the end of iteration  $i$  if she observed less than  $0.001p_e L_i$  erasures in each of the parts of iteration  $i$ , and Bob's transmissions at the second part contains more 0's than 1's (i.e., it decodes to the success string rather than to the error string). Alice gives as an output the same output that  $\pi'$  has generated in the iteration in which she terminated. Bob terminates at the end of iteration  $j$  if he observed less than  $0.001p_e L_j$  erasures in the first part of  $j$  and at most  $0.001p_e L_j$  of the received bits in the first part of  $j$  are 1's. Bob gives as an output the output of the latest iteration  $k$ , with  $k < j$ , in which (1) he observed less than  $0.001p_e L_j$  erasures in the first part and (2) he received at least  $L_k/40$  ones from Alice in the first part. We call a *valid iteration* any iteration that satisfies these two conditions.

## 4.1 Analysis

In this section we prove the following theorem.

► **Theorem 4.1.** *Given any two-party binary interactive protocol  $\pi$  of length  $N$ , there exists an efficient randomized protocol  $\Pi$  of length  $O(N + T)$  that simulates  $\pi$  with probability  $1 - 2^{-\Omega(N)}$  over a binary channel in the presence of an arbitrary and a priori unknown number  $T$  of mUPEF corruptions, with  $p_i = 2/3$  for all  $i$ .*

We start with proving the correctness of our coding scheme: we begin by demonstrating that Alice's output is correct with high probability. Additionally, we show that Bob's output at Alice's termination is also correct. Then, we prove that Bob terminates after Alice, that Alice terminates in a *valid* iteration, and that there are no valid iterations afterwards. This would imply that Bob gives the right output as well.

Recall that  $\pi'$  is resilient to 0.1-fraction of substitutions. In addition, recall the padding mechanism; in order to cause a bit substitution in  $\pi'$  in some iteration  $i$ , at least a half of its  $2^i$  transmitted copies must be flipped or erased. Thus, if there are less than  $L_i/20$  corruptions during the first part of iteration  $i$  (that is, indices that the adversary puts in  $E$ ), then  $\pi'$  must give the correct output at the end of iteration  $i$ , for both Alice and Bob.

In the following lemma, we show that if there are *more* than  $L_i/20$  corruptions in some iteration  $i$ , then Alice continues to executing iteration  $i + 1$  and does not terminate at the end of iteration  $i$ , except with a negligible probability of  $2^{-\Omega_{p_e}(L_i)}$ .

► **Lemma 4.2.** *Assume that in the first part of iteration  $i$  there are  $c_i \geq L_i/20$  corruptions. Then, the probability that Alice terminates at the end of iteration  $i$  is  $2^{-\Omega_{p_e}(L_i)}$ .*

## 43:12 Interactive Coding with Unbounded Noise

**Proof.** We divide the proof into two separate cases: (1) each of Alice and Bob observes less than  $0.001p_e L_i$  erasures in the first part, and (2) Bob observes more than  $0.001p_e L_i$  erasures but Alice does not. Of course, if more than  $0.001p_e L_i$  erasures are observed by Alice during the first part, she does not terminate by definition.

First, we find the probability of case (1) to occur. Denote by  $E_i$  the subset of the noise pattern  $E$  containing only rounds in the first part of iteration  $i$ . Then,  $|E_i| \geq c_i \geq L_i/20$ . Let  $e'$  be the number of rounds during the first part of iteration  $i$  that were erased because the event of channel erasure, which happens with probability  $p_e$ , occurred. It holds that  $\mathbb{E}[e'] = |E_i|p_e \geq 0.05p_e L_i$ , and by Chernoff's inequality (Theorem 4.4(2) in [24]),

$$\Pr(e' < 0.002p_e L_i) \leq \Pr(e' < 0.04\mathbb{E}[e']) \leq e^{-\mathbb{E}[e'] \frac{0.96^2}{2}} \leq e^{-0.05p_e L_i \frac{0.96^2}{2}} \in 2^{-\Omega_{p_e}(L_i)}.$$

Thus, if  $c_i \geq L_i/20$ , then  $e' < 0.002p_e L_i$  with probability  $2^{-\Omega_{p_e}(L_i)}$ , which bounds the probability of case (1) to occur.

In case (2), Bob sees a lot of erasures and sends the error string. In order for Alice to terminate, it is necessary for her to decode this message as the success string. For this to happen, it is necessary that the adversary corrupts at least  $L_i/2$  bits during the second part of the  $i$ -th iteration. Similar to the proof of case (1), the adversary succeeds to corrupt so many bits while causing less than  $0.001p_e L_i$  erasures during the second part with probability of at most  $2^{-\Omega_{p_e}(L_i)}$ . We conclude that Alice terminates at the end of iteration  $i$  with probability of at most  $2^{-\Omega_{p_e}(L_i)}$ . ◀

The above lemma indicates that, in the event of an excessive number of corruptions, Alice will not terminate. The following observation complements this idea and states that if Alice does terminate, the computation of  $\pi'$  is successful with high probability, hence, her output in  $\Pi$  is correct.

► **Observation 4.3.** *When Alice terminates,  $\pi'$  gives the correct output, with probability  $1 - 2^{-\Omega_{p_e}(N)}$ .*

**Proof.** If during a given iteration there were less than  $L_i/20$  corruptions, then the resilience of  $\pi'$  guarantees that it gives the right output, and Alice terminates. The event in which Alice terminates and provides incorrect output can only occur when there are more corruptions in a specific iteration, the probability of which was constrained in Lemma 4.2. A union bound over all possible iterations (while recalling that  $L_i = |\pi'|2^i$ ) bounds the probability for Alice to give an incorrect output, by

$$\sum_{i=0}^{\infty} 2^{-\Omega_{p_e}(L_i)} = 2^{-\Omega_{p_e}(|\pi'|)} = 2^{-\Omega_{p_e}(N)}. \quad \blacktriangleleft$$

Next, we prove that Bob terminates only after Alice has already terminated, with high probability.

► **Lemma 4.4.** *Consider an iteration  $i$  in which Alice has not yet terminated. Then, the probability that Bob terminates at the end of iteration  $i$  is at most  $2^{-\Omega_{p_e}(L_i)}$ .*

**Proof.** In order to terminate at the end of iteration  $i$ , Bob must observe less than  $0.001p_e L_i$  erasures in the first part of the iteration. In a manner analogous to the argument presented in Lemma 4.2, it can be shown that, with a probability approaching  $1 - 2^{-\Omega_{p_e}(L_i)}$ , there will be a total of less than  $L_i/20$  corrupted messages received by Bob during the first part of iteration  $i$ . Recall that as long as Alice has not terminated, at least  $L_i/8$  out of the  $L_i/2$

bits she sends in iteration  $i$  are ones. Consider these transmissions only. Even if all the corruptions during the first part of  $i$  occur during these transmissions, then Bob will observe at least  $0.001p_e L_i$  ones in the first part of  $i$ , and will therefore not terminate by definition. Consequently, Bob terminates in iteration  $i$  with probability  $2^{-\Omega_{p_e}(L_i)}$ . ◀

Bob's output is the output of  $\pi'$  in the last *valid* iteration prior to his termination iteration. The following lemma shows that, with high probability, the last valid iteration is the same iteration in which Alice has terminated. By Observation 4.3,  $\pi'$  gives the correct output in that iteration.

► **Lemma 4.5.** *Denote by  $i$  the iteration in which Alice terminates. Then,  $i$  is a valid iteration with probability  $1 - 2^{-\Omega_{p_e}(L_i)}$ . Further, the probability that there is a valid iteration  $j > i$  is  $2^{-\Omega_{p_e}(N)}$ .*

**Proof.** In order to prove that  $i$  is a valid iteration, we have to show that Bob observes less than  $0.001p_e L_i$  erasures in the first part of  $i$ , and that he receives at least  $L_i/40$  ones from Alice in the first part.

First, note that Bob observes less than  $0.001p_e L_i$  erasures in the first part of  $i$  if and only if he sends the success string in the second part. We demonstrate that with probability  $1 - 2^{-\Omega_{p_e}(L_i)}$  this is the case. As illustrated in case (2) of Lemma 4.2, when Bob transmits the error string to Alice in the second part of  $i$ , Alice terminates with probability  $2^{-\Omega_{p_e}(L_i)}$ . Since Alice terminates in  $i$ , there is a probability of  $2^{-\Omega_{p_e}(L_i)}$  that Bob sends the error string in the second part of  $i$  and observes more than  $0.001p_e L_i$  erasures in the first part of  $i$ .

We proceed to show that Bob receives at least  $L_i/40$  ones from Alice in the first part of  $i$ . By Lemma 4.2, during the first part of iteration  $i$  there are less than  $L_i/20$  corruptions with probability  $1 - 2^{-\Omega_{p_e}(L_i)}$ . Additionally, Alice always sends at least  $L_i/8$  ones in the first part. Thus, Bob receives at least  $L_i/40$  ones during the first part of  $i$  with probability  $1 - 2^{-\Omega_{p_e}(L_i)}$ , and this is the probability of  $i$  to be a valid iteration.

Let  $j$  be an iteration such that  $j > i$ . Recall that after Alice terminates the channel sends Bob zeros, by default. We show that  $j$  is not a valid iteration with high probability, by dividing into two cases. If there are at least  $L_j/40$  flips in the transmissions towards Bob during the first part of iteration  $j$ , then with probability  $1 - 2^{-\Omega_{p_e}(L_j)}$  Bob observes at least  $0.001p_e L_j$  erasures, similar to case (1) in Lemma 4.2, thus  $j$  is not valid. If there are less than  $L_j/40$  flips in these transmissions, then Bob receives less than  $L_j/40$  ones and  $j$  is not a valid iteration either. Thus, iteration  $j$  is valid with probability of at most  $2^{-\Omega_{p_e}(L_j)}$ . Since  $L_j = |\pi'|2^j$ , applying a union bound on all the iterations gives a probability of  $\sum_{j=i}^{\infty} 2^{-\Omega_{p_e}(L_j)} = 2^{-\Omega_{p_e}(N)}$  to the event that there is a valid iteration after  $i$ . ◀

We may use a union bound to conclude the correctness part for Bob. The overall probability of Bob to terminate after Alice is  $1 - 2^{-\Omega_{p_e}(N)}$ . The probability of Bob to declare Alice's termination iteration as valid is  $1 - 2^{-\Omega_{p_e}(N)}$ . Bob gives the correct output at this iteration with probability  $1 - 2^{-\Omega_{p_e}(N)}$  by Lemma 4.2, and this iteration is the last valid iteration with probability  $1 - 2^{-\Omega_{p_e}(N)}$ . We may use the inclusion-exclusion principle to show that the probability of the intersection of all these four events is  $1 - 2^{-\Omega_{p_e}(N)}$ . Recall that for any  $A, B$  it holds that  $1 \geq P(A \cup B) = P(A) + P(B) - P(A \cap B)$ . Then, the fact that  $P(A), P(B) \in 1 - 2^{-\Omega_{p_e}(N)}$  implies that  $P(A \cap B) \geq 2(1 - 2^{-\Omega_{p_e}(N)}) - 1 \in 1 - 2^{-\Omega_{p_e}(N)}$ . Thus, the probability of the event in which Bob gives the correct output at his termination is  $1 - 2^{-\Omega_{p_e}(N)}$ , and we have completed the correctness part of Theorem 4.1.

The communication complexity of  $\Pi$  is  $\sum_{i=1}^{i_B} 2L_i$ , with  $i_B$  being the iteration in which Bob terminates. At any iteration  $i$  before Alice terminates, the adversary has to select at least  $0.001p_e L_i$  bits to corrupt in order to prevent Alice from terminating. In addition,

at any iteration  $i$  after Alice's and before Bob's terminations, the adversary has to select at least  $0.001p_e L_i$  bits to corrupt in order to prevent Bob from terminating. Denote the iteration in which Alice terminates by  $i_A < i_B$ . Then,  $\sum_{i=1, i \neq i_A}^{i_B-1} 0.001p_e L_i < T$ , so  $\sum_{i=1, i \neq i_A}^{i_B-1} 2L_i < 2000/p_e \times T$ . Since  $L_i$  satisfies  $L_i = 2L_{i-1}$  for all  $i$ , and since  $L_{i_A-1}, L_{i_B-1}$  are included in  $\sum_{i=1, i \neq i_A}^{i_B-1} 2L_i < 2000/p_e \times T$ , then  $\sum_{i=1}^{i_B} 2L_i < 6000/p_e \times T$  and the communication complexity is  $O_{p_e}(N + T)$ .

## 4.2 Obtaining a UF-optimal coding scheme

In this section we construct a UF-model scheme based on the mUPEF protocol  $\Pi$ , while maintaining a communication complexity of  $O(N + T)$ . Recall, we set  $p_e = 1/3$ . This means that adversarial corruption in the  $i$ -th transmission of  $\Pi$ , becomes detectable (an erasure) with probability at least  $1/3$ . We would like to simulate this property in the UF model.

Towards this goal, we independently encode each bit of  $\Pi$  using a random code of length 5. In particular, we encode a 0 to one of  $\{00000, 10000, 01000\}$  with equal probability, and encode a 1 to one of  $\{00100, 10010, 01001\}$  with equal probability. A received 5-bit word is decoded to a 0 or a 1 only if it belongs to respective set, or otherwise it is considered as an erasure. It can easily be seen that any pattern of 1 to 5 bit-flips decodes to an erasure with probability at least  $1/3$  as desired. We have thus inflated the scheme by only a constant factor. The bit complexity of the resulting UF scheme is then  $5 \cdot |\Pi| = O(N + T)$ .

---

### References

- 1 Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. A scalable algorithm for multiparty interactive communication with private channels. In *ICDCN 2020: 21st International Conference on Distributed Computing and Networking, Kolkata, India, January 4-7, 2020*, ICDCN '20, pages 8:1–8:15, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3369740.3369771.
- 2 Zvika Brakerski, Yael T. Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35:1–35:30, December 2014. doi:10.1145/2661628.
- 3 Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. *SIAM Journal on Computing*, 46(1):388–428, 2017. doi:10.1137/141002001.
- 4 Mark Braverman, Ran Gelles, Jieming Mao, and Rafail Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, October 2017. doi:10.1109/TIT.2017.2734881.
- 5 Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, November 2014. doi:10.1109/TIT.2014.2353994.
- 6 Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965, pages 471–488, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-78967-3\_27.
- 7 Varsha Dani, Thomas P. Hayes, Mahnush Movahedi, Jared Saia, and Maxwell Young. Interactive communication with unknown noise rate. *Information and Computation*, 261:464–486, 2018. doi:10.1016/j.ic.2018.02.018.
- 8 Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. *IEEE Transactions on Information Theory*, 62(8):4575–4588, August 2016. doi:10.1109/TIT.2016.2582176.

- 9 Klim Efremenko, Elad Haramaty, and Yael Tauman Kalai. Interactive coding with constant round and communication blowup. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:34, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2020.7.
- 10 Eden Fargion, Ran Gelles, and Meghal Gupta. Interactive coding with unbounded noise, 2024. arXiv:2407.09463.
- 11 Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. *IEEE Transactions on Information Theory*, 61(1):133–145, January 2015. doi:10.1109/TIT.2014.2367094.
- 12 Ran Gelles. Coding for interactive communication: A survey. *Foundations and Trends® in Theoretical Computer Science*, 13(1-2):1–157, 2017. doi:10.1561/04000000079.
- 13 Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. *SIAM Journal on Computing*, 46(4):1449–1472, 2017. doi:10.1137/15M1052202.
- 14 Ran Gelles, Bernhard Haeupler, Gillat Kol, Noga Ron-Zewi, and Avi Wigderson. Explicit capacity approaching coding for interactive communication. *IEEE Transactions on Information Theory*, 64(10):6546–6560, October 2018. doi:10.1109/TIT.2018.2829764.
- 15 Ran Gelles and Siddharth Iyer. Interactive coding resilient to an unknown number of erasures. In *23rd International Conference on Principles of Distributed Systems, OPODIS 2019, December 17-19, 2019, Neuchâtel, Switzerland*, volume 153 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:16, 2019. doi:10.4230/LIPIcs.OPODIS.2019.13.
- 16 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding – part I: oblivious insertions, deletions and substitutions. *IEEE Transactions on Information Theory*, 67(6):3411–3437, 2021. doi:10.1109/TIT.2021.3066009.
- 17 Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding - part II: non-oblivious noise. *IEEE Transactions on Information Theory*, 68(7):4723–4749, 2022. doi:10.1109/TIT.2022.3160515.
- 18 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, March 2014. doi:10.1109/TIT.2013.2294186.
- 19 Mohsen Ghaffari and Bernhard Haeupler. Optimal error rates for interactive coding II: efficiency and list decoding. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 394–403. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.49.
- 20 Meghal Gupta and Rachel Yun Zhang. The optimal error resilience of interactive communication over binary channels. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, STOC 2022, pages 948–961, 2022. doi:10.1145/3519935.3519985.
- 21 Bernhard Haeupler. Interactive channel capacity revisited. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 226–235, 2014. doi:10.1109/FOCS.2014.32.
- 22 Bernhard Haeupler, Amirbehshad Shahrashbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.
- 23 Gillat Kol and Ran Raz. Interactive channel capacity. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 715–724, 2013. doi:10.1145/2488608.2488699.

## 43:16 Interactive Coding with Unbounded Noise

- 24 Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.
- 25 Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 724–733, Los Alamitos, CA, USA, 1992. IEEE Computer Society. doi:10.1109/SFCS.1992.267778.
- 26 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, November 1996. doi:10.1109/18.556671.
- 27 Alexander A. Sherstov and Pei Wu. Optimal interactive coding for insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 65(10):5971–6000, October 2019. doi:10.1109/TIT.2019.2916927.

# Optimal Pseudorandom Generators for Low-Degree Polynomials over Moderately Large Fields

Ashish Dwivedi ✉ 🏠 

Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA

Zeyu Guo ✉ 🏠 

Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA

Ben Lee Volk ✉ 🏠 

Efi Arazi School of Computer Science, Reichman University, Israel

---

## Abstract

We construct explicit pseudorandom generators that fool  $n$ -variate polynomials of degree at most  $d$  over a finite field  $\mathbb{F}_q$ . The seed length of our generators is  $O(d \log n + \log q)$ , over fields of size exponential in  $d$  and characteristic at least  $d(d-1)+1$ . Previous constructions such as Bogdanov’s (STOC 2005) and Derksen and Viola’s (FOCS 2022) had either suboptimal seed length or required the field size to depend on  $n$ .

Our approach follows Bogdanov’s paradigm while incorporating techniques from Lecerf’s factorization algorithm (J. Symb. Comput. 2007) and insights from the construction of Derksen and Viola regarding the role of indecomposability of polynomials.

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

**Keywords and phrases** Pseudorandom Generators, Low Degree Polynomials

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.44

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2402.11915>

**Funding** *Ben Lee Volk*: The research leading to these results has received funding from the Israel Science Foundation (grant number 843/23).

**Acknowledgements** We thank Jesse Goodman and Pooya Hatami for helpful discussions. Part of this work was carried out while the first two authors were visiting the Simons Institute for the Theory of Computing at UC Berkeley. We thank the institute for its support and hospitality.

## 1 Introduction

The role of randomness in efficient computation is one of the central topics in complexity theory: random bits are useful for designing algorithms, but producing random bits comes at a cost and it is often desirable to reduce them or eliminate them altogether. One of the simplest yet most profound insights in this area is that efficient algorithms are, by definition, computationally limited, and cannot perform arbitrary statistical tests over their random bits. Therefore, one may hope to construct *pseudorandom* distributions that use less random bits but are able to “fool” some limited classes of tests, that cannot distinguish between them and between truly random bits.

For the pseudorandom distributions to be useful, they need to be efficiently computable themselves. This is usually modeled as a *pseudorandom generator* (PRG, for short). A PRG for a class of  $\mathcal{C}$  is an efficiently computable function  $G : S \rightarrow B$  such that for every function  $f \in \mathcal{C}$ , the distributions  $f(\mathbf{U}_B)$  and  $f(G(\mathbf{U}_S))$  are close in statistical distance, where  $\mathbf{U}_A$  denotes the uniform distribution over the set  $A$ . Namely, the two experiments of applying  $f(\cdot)$  to a uniformly random element of  $B$ , and applying  $f(G(\cdot))$  to a uniformly random element



© Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 44; pp. 44:1–44:19



Leibniz International Proceedings in Informatics

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

of  $S$ , give roughly the same results. For this to be useful and non-trivial, obviously the set  $S$  needs to be significantly smaller than  $B$ . The quantity  $\log |S|$  is called the *seed length* of the generator.

There has been a significant amount of work on constructing pseudorandom generators for various types of restricted distinguishers. In its most general form, where the distinguisher is allowed to be an arbitrary efficient (even non-uniform) algorithm, constructing such PRGs would imply breakthrough lower bounds in complexity theory. However, there are also unconditional constructions of PRGs for distinguishers coming from certain smaller complexity classes (see, for example, the surveys [34, 17]).

In this paper, we focus on pseudorandom generators in the algebraic setting. Here, the restriction on the distinguishers is of algebraic nature: we seek to fool distinguishers that are low-degree  $n$ -variate polynomials over finite fields.

The problem of fooling low-degree polynomials is well-studied. The most basic case is polynomials of degree one, i.e., fooling linear functions. Such generators are also known as  $\varepsilon$ -biased sets, and this problem was traditionally studied over  $\mathbb{F}_2$ , although some of the constructions can be generalized to larger fields. This concept was first defined and considered by Naor and Naor [24], with various improved constructions given by [2, 14, 4], culminating in a recent nearly-optimal construction by Ta-Shma [33]. The seed length in those constructions is  $O(\log n + \log q + \log(1/\varepsilon))$ , where  $n$  denotes the number of variables,  $\varepsilon$  the error of the PRG, and  $q$  the field size.

While focusing on polynomials of degree one might seem a bit too restrictive,  $\varepsilon$ -biased sets have found numerous applications throughout the field of pseudorandomness and derandomization, and in the theory of computation in general.

One example relevant to this work is that  $\varepsilon$ -biased sets are in fact a basic building block in a construction of PRGs for higher-degree polynomials, using a paradigm initiated by Bogdanov and Viola [7]. They suggested constructing a generator for degree- $d$  polynomials by summing up  $\ell = \ell(d)$  independent copies of a generator for degree-one polynomials. The paper [7] proved a conditional result when the number of summands is  $d$ , assuming certain additive combinatorics conjectures. Lovett [22] showed how to prove an unconditional result at the cost of making the number of summands  $2^d$ . Finally, Viola [35] showed (unconditionally) that in fact  $d$  summands suffice. The seed length in his construction is  $O(d \log n + d2^d \log(q/\varepsilon))$ . Indeed, even though the construction only sums  $d$  copies of a generator for degree-one polynomials, for the analysis to go through, the error of this generator needs to be as small as  $\varepsilon^{2^d}$  (for the final error of the generator for degree- $d$  polynomials to be  $\varepsilon$ ), which incurs a factor of  $2^d$  in the final seed length. Improving this generator and in particular obtaining meaningful results for polynomials of degree greater than  $\log n$  is an extremely important open problem in complexity theory. One reason is that such pseudorandom generators will yield pseudorandom generators for small constant-depth circuits with parity gates, since Razborov [25] and Smolensky [32] famously proved that functions computed by such circuits are approximated by low-degree polynomials.

All the constructions mentioned above work for any field. There are, however, better results when the field size  $q$  is assumed to be large (typically, at least polynomially large in  $d$  and  $1/\varepsilon$ ). This assumption is useful since it allows one to use powerful tools from algebraic geometry, such as Weil-type estimates [36] on the number of points of varieties over finite fields.

This line of work was initiated by Bogdanov [6], who showed how to use different pseudorandom objects called *hitting set generators* for low-degree polynomials in order to construct pseudorandom generators. Bogdanov's work, followed by the later improved



constructions of hitting set generators [19, 23, 10, 16], resulted in a PRG with seed length  $O(d^4 \log n + \log q)$  assuming  $q \geq Cd^6/\varepsilon^2$  for a sufficiently large constant  $C$ .<sup>1</sup> Over fields of large enough characteristic, combining Bogdanov’s paradigm with the results of Lecerf [21] results in improving the  $d^4$  factor to  $d^2$ . We expand more on Bogdanov’s and Lecerf’s techniques in Section 1.2, as both are very relevant to this work.

More recently, Derksen and Viola [12] introduced fundamentally new techniques for this problem, with tools coming from invariant theory. One of their key ideas is to construct a low-degree polynomial map on a few variables that preserves the *indecomposability* property of a polynomial  $f$  when composed with it (we refer to Section 2 for more on that). Using this new tool in conjunction with other techniques, they are able to construct generators with seed length  $O(d \log(dn) + \log q)$ , assuming  $q \geq Cd^4n^\delta/\varepsilon^2$  (for some large constant  $C$  and small constant  $\delta$ ), or seed length  $O(d \log n \log(d \log n) + \log q)$  for  $q \geq C(d \log n)^4/\varepsilon^2$ . One should note, however, that the optimal parameters in the construction of [12] are obtained after composing their construction with Bogdanov’s original construction.

A related natural question is how small the seed length can potentially be. Alon, Ben-Eliezer and Krivelevich [1] considered this question and proved a lower bound of  $\Omega(d \log(n/d) + \log q + \log(1/\varepsilon))$  on the seed length. Thus, we see that the explicit constructions of [12] come very close to the optimal bound. However, unlike the result of Bogdanov [6], in the construction of Derksen and Viola [12] the minimum field size depends on the number of variables  $n$ .

### 1.1 Our Results

In this paper, we provide an improved construction of PRGs for low-degree polynomials, with an even shorter seed length, assuming the field size is exponentially large in  $d$  (but independent of  $n$ ).

► **Theorem 1.1.** Let  $\mathbb{F}_q$  be a finite field of characteristic at least  $d(d-1)+1$  and size  $q \geq C(d2^d/\varepsilon + d^4/\varepsilon^2)$  (for some sufficiently large absolute constant  $C$ ). Then, there exists an explicit pseudorandom generator that fools  $n$ -variate polynomials of degree at most  $d$  over  $\mathbb{F}_q$  with error  $\varepsilon$  and seed length  $O(d \log n + \log q)$ .

For convenience, we summarize the comparison between Theorem 1.1 and the results of Bogdanov [6], Viola [35] and Derksen and Viola [12] in the following table. All the entries in this table are given up to some constant factors, but for ease of readability, we omit  $O(\cdot)$  notations.

|               | Seed Length                                  | Field Size                             | Characteristic  |
|---------------|--|--|-----------------|
| [35]          | $d \log n + d \cdot 2^d \log(q/\varepsilon)$ | Every $q \geq 2$                       | Any             |
| [6]+[16]      | $d^4 \log n + \log q$                        | $d^6/\varepsilon^2$                    | Any             |
| [6]+[16]+[21] | $d^2 \log n + \log q$                        | $d^6/\varepsilon^2$                    | $\geq d(d-1)+1$ |
| [12]          | $d \log(n) + \log q$                         | $d^4 n^{0.001}/\varepsilon^2$          | Any             |
| [12]          | $d \log n \cdot \log(d \log n) + \log q$     | $(d \log n)^4/\varepsilon^2$           | Any             |
| This paper:   | $d \log n + \log q$                          | $d2^d/\varepsilon + d^4/\varepsilon^2$ | $\geq d(d-1)+1$ |

We also prove that, if we only want to fool polynomials of *prime* degree up to  $d$ , then the required field size in Theorem 1.1 can be improved to  $O(d^4/\varepsilon^2)$ , avoiding an exponential dependence on  $d$ .

<sup>1</sup> One should note that since  $q$  is polynomially large in  $1/\varepsilon$ , the seed length also implicitly depends on  $\log(1/\varepsilon)$  through the  $\log q$  term.

► **Theorem 1.2.** Let  $\mathbb{F}_q$  be a finite field of characteristic at least  $d(d-1)+1$  and size  $q \geq C(d^4/\varepsilon^2)$  (for some sufficiently large absolute constant  $C$ ). Then, there exists an explicit pseudorandom generator that fools  $n$ -variate polynomials of prime degree up to  $d$  over  $\mathbb{F}_q$  with error  $\varepsilon$  and seed length  $O(d \log n + \log q)$ .

## 1.2 Proof Techniques

We start by reviewing the proof of Bogdanov’s PRG. Bogdanov’s idea is to consider restrictions of the polynomial  $f$  we are trying to fool onto planes, and to argue that “most” planes preserve the output distribution of the polynomial. Since a plane is a two-dimensional subspace, after having selected a good plane, we only need to sample two more field elements to select a random element from the plane.

The question now is how to find a good plane. Here, Bogdanov uses results by Kaltofen [18], who proved an effective version of Hilbert’s irreducibility theorem. Kaltofen demonstrated that, for every degree- $d$  irreducible polynomial  $f$ , there exists a polynomial  $P$  of degree roughly  $d^4$ , whose variables correspond to the parameters of the plane, such that every point at which  $P$  is nonzero corresponds to a “good” plane for  $f$  – that is, a plane that preserves the irreducibility of  $f$ . Therefore, we can use a hitting set generator for polynomials of degree  $d^4$  to find a good plane. This results in a factor of  $d^4$  in the final seed length.

Over fields of large characteristic (or characteristic zero), Lecerf [20, 21] obtained an improved upper bound of  $O(d^2)$  on the degree of such polynomials, based on ideas of Ruppert [26, 27] and Gao [15]. However, Lecerf also presents an example where the degree of such a polynomial must be at least  $\Omega(d^2)$ , demonstrating that this approach alone may not suffice to achieve an improvement within Bogdanov’s framework.

Derksen and Viola circumvent this problem by using a different approach: rather than preserving irreducibility (or more generality, if a polynomial is reducible, preserving its number of irreducible factors) as in Bogdanov’s approach, they construct a map that preserves the *indecomposability* of polynomials: a polynomial  $f(x_1, \dots, x_n)$  is indecomposable if it cannot be written as  $f = g(h(x_1, \dots, x_n))$  where  $g$  is a univariate polynomial of degree at least 2. Over large fields, irreducibility and indecomposability both guarantee that the polynomial is roughly equidistributed, and in fact these notions are tightly connected. We refer to Section 2 for the precise definitions and description of those connections.

To prove our result, we revisit Bogdanov’s approach, while noting that by applying Lecerf’s results [21] (rather than Kaltofen’s bounds [18]) in a more careful way, and assuming the field is sufficiently large, it is in fact enough to use hitting sets generators only for polynomials of degree  $O(d)$ , rather than  $O(d^2)$ . We also incorporate the insights from [12] regarding indecomposability and equidistribution. This requires following the outline above but making sure that at each step, we only need to hit polynomials of degree  $O(d)$ .

Following Lecerf’s notation and terminology, suppose  $F(x_1, \dots, x_n, y)$  is an irreducible polynomial of degree at most  $d$ . A point  $\mathbf{a} = (a_1, \dots, a_n)$  is called a *Bertinian good* point if the bivariate polynomial  $H(x, y) = F(a_1x, \dots, a_nx, y)$  remains irreducible. Lecerf proves that there exists a polynomial  $\mathcal{A}(z_1, \dots, z_n)$  of degree  $O(d^2)$  such that if  $\mathcal{A}(a_1, \dots, a_n) \neq 0$  then  $\mathbf{a}$  is a Bertinian good point. This is achieved by transforming the question of irreducibility into a question about the rank of a solution space for a certain linear system that depends on  $a_1, \dots, a_n$  (see Section 4). This transformation naturally leads to defining  $\mathcal{A}$  as a certain minor of the matrix representing that linear system. The minor has dimensions  $O(d) \times O(d)$ , and each entry of the matrix is a polynomial of degree  $O(d)$ , which results in a total bound of  $O(d^2)$  on the degree of its determinant,  $\mathcal{A}$ .

We, however, observe that Lecerf’s results actually imply a much stronger structure of this linear system: its solution space, over any field, is always spanned by vectors whose entries are in  $\{0, 1\}$ . In fact, Lecerf directly characterizes the relationship between the irreducible factors of  $H(x, y)$  and the vectors spanning the solution space, though this detail is irrelevant for the moment.

Thus, in Lecerf’s argument,  $a_1, \dots, a_n$  are chosen such that a particular minor is nonzero, namely, a certain linear system has no non-trivial solutions. But it is enough to select  $a_1, \dots, a_n$  in a way that only guarantees that this linear system has no non-trivial  $0/1$  solutions!

Fixing any vector  $u \in \{0, 1\}^d$ , the requirement that  $u$  is not a solution to the linear system turns out to be a condition expressible as  $u$  being a nonzero of a polynomial of degree  $O(d)$ , rather than  $O(d^2)$ . If we pick  $a_1, \dots, a_n$  from a hitting set generator with error  $\delta$  smaller than  $2^{-d}$ , we can afford to take a union bound over all vectors in  $\{0, 1\}^d$  and ensure that none of them is a solution to the linear system while keeping the total error small. This is not a big price to pay in terms of the seed length of the HSG, which is  $O(d \log n + \log(1/\delta))$ , so requiring  $\delta$  to be exponentially small in  $d$  adds an insignificant additive  $O(d)$  term. Where we do pay the price for the small error is in the field size, since the explicit construction of the HSG we use requires the field size to be at least roughly  $d/\delta$ . Fortunately, however, the dependence of the seed length of our generator on the field size  $q$  is also by an additive  $O(\log q)$  term, which means that once more requiring  $q$  to be exponentially large in  $d$  has no adverse effects even on the total seed length of the PRG.

We briefly remark that, for technical reasons, Lecerf’s result also requires the characteristic of the underlying field to be zero or at least  $d(d-1)+1$ . We further elaborate on Lecerf’s techniques in Section 4.

We finally mention another important technical point. Lecerf’s irreducibility characterization [21] assumes a technical condition on the polynomial, which he called Hypothesis (H) (see Section 3). Such a “preprocessing” step, which makes the polynomial monic in a certain distinguished variable, is common to many factorization algorithms, and can usually be easily guaranteed by applying a random linear transformation to the variables. However, doing this in the naïve way would require the use of too many random bits. To solve this problem, in Section 3 we show that this part can also be derandomized by using a hitting set generator for polynomials of degree  $O(d)$ . As explained in Section 3, this part also uses crucially indecomposability (rather than irreducibility) in a novel way.

## 2 Preliminaries

We now define the basic objects studied in this paper and introduce the fundamental mathematical concepts used.

### Notation

All logarithms are base 2. Denote by  $\mathbb{N}$  the set of natural numbers  $\{0, 1, 2, \dots\}$ . For  $n \in \mathbb{N}$ , define  $[n] = \{1, 2, \dots, n\}$ . For a finite set  $A$ , denote by  $\mathbf{U}_A$  the uniform distribution over  $A$ .

We often use symbols in bold, e.g.,  $\mathbf{a}$  or  $\mathbf{x}$ , as the shorthand for a vector  $(a_1, \dots, a_n)$  or a sequence of variables  $x_1, \dots, x_n$ .

Denote by  $\mathbb{F}_q$  the finite field of size  $q$ . The algebraic closure of a field  $\mathbb{F}$  is denoted by  $\overline{\mathbb{F}}$ . For a commutative ring  $A$  and variables  $x_1, \dots, x_n$ , we denote by  $A[[x_1, \dots, x_n]]$  or  $A[[\mathbf{x}]]$  the *ring of formal power series* over  $A$  in  $x_1, \dots, x_n$ , i.e.,

$$A[[\mathbf{x}]] = \left\{ \sum_{\mathbf{e}=(e_1, \dots, e_n) \in \mathbb{N}^n} a_{\mathbf{e}} x_1^{e_1} \cdots x_n^{e_n} : a_{\mathbf{e}} \in A \right\}.$$

### Pseudorandom Generators and Hitting Set Generators

► **Definition 2.1** (Pseudorandom generator, PRG). Let  $\mathbb{F}_q$  be a finite field. A *pseudorandom generator* (PRG) for  $n$ -variate polynomials of degree at most  $d$  over  $\mathbb{F}_q$  with error  $\varepsilon$  is an efficiently computable map  $G : S \rightarrow \mathbb{F}_q^n$  from a finite set  $S \neq \emptyset$  such that for every such polynomial  $f$  of degree at most  $d$ , the two distributions  $f(G(\mathbf{U}_S))$  and  $f(\mathbf{U}_{\mathbb{F}_q^n})$  are  $\varepsilon$ -close in statistical distance. That is,

$$\frac{1}{2} \sum_{a \in \mathbb{F}_q} \left| \Pr_{\mathbf{x} \in \mathbb{F}_q^n} [f(\mathbf{x}) = a] - \Pr_{\mathbf{y} \in S} [f(G(\mathbf{y})) = a] \right| \leq \varepsilon.$$

The quantity  $\log |S|$  is called the *seed length* of  $G$ .

A weaker object than a PRG is a *hitting set generator*. Here, we only require that a nonzero polynomial is nonzero (with high probability) on the output of the generator.

► **Definition 2.2** (Hitting set generator, HSG). Let  $\mathbb{F}$  be a field. A *hitting set generator* (HSG) with density  $1 - \delta$  for  $n$ -variate polynomials of degree at most  $d$  over  $\mathbb{F}$  is an efficiently computable map  $H : S \rightarrow \mathbb{F}^n$  from a finite set  $S \neq \emptyset$  such that for every such nonzero polynomial  $f$  of degree at most  $d$ ,

$$\Pr_{\mathbf{y} \in S} [f(H(\mathbf{y})) = 0] \leq \delta.$$

The quantity  $\log |S|$  is called the *seed length* of  $G$ .

Building on the earlier work [19, 23] and algebraic-geometric codes, Guruswami and Xing [16] constructed explicit HSGs for low-degree polynomials with asymptotically optimal seed length and density.

► **Theorem 2.3** ([16]). There exists an absolute constant  $C$  such that for any  $n, d, q, \delta$ , such that  $q \geq Cd/\delta$ , there exists an explicit HSG for  $n$ -variate polynomials of degree at most  $d$  over  $\mathbb{F}_q$  with density  $1 - \delta$  and seed length  $O(d \log n + \log(1/\delta))$ .

It should also be noted that for hitting set generators, the field  $\mathbb{F}$  does not have to be finite. This generality is used in the statement of the following fact, that an HSG for a field  $\mathbb{F}$  is also a HSG for any extension field  $\mathbb{K}$  of  $\mathbb{F}$ .

► **Fact 2.4** ([6, 12]). Let  $H : S \rightarrow \mathbb{F}$  be an HSG with density  $1 - \delta$  for polynomials of degree at most  $d$  over a field  $\mathbb{F}$ , and let  $\mathbb{K}$  be an extension of  $\mathbb{F}$ . Then  $H$  is also an HSG with density  $1 - \delta$  for polynomials of degree at most  $d$  over  $\mathbb{K}$ .

**Proof.** Let  $\mathcal{B}$  be a basis of  $\mathbb{K}$  over  $\mathbb{F}$ , and let  $f$  be a nonzero polynomial in  $\mathbb{K}[x_1, \dots, x_n]$  of degree at most  $d$ . By expressing every coefficient  $c \in \mathbb{K}$  of a monomial in  $f$  as a linear combination  $c = \sum_{b \in \mathcal{B}} a_b \cdot b$  with  $a_b \in \mathbb{F}$  for every  $b \in \mathcal{B}$ , we may write  $f = \sum_{b \in \mathcal{B}} f_b \cdot b$  such that  $f_b \in \mathbb{F}[x_1, \dots, x_n]$  is a polynomial of degree at most  $d$  for every  $b \in \mathcal{B}$ , and at least one  $f_b$  is nonzero. Thus, for any  $u \in S$ ,  $f(H(u)) = \sum_{b \in \mathcal{B}} f_b(u) \cdot b$  is nonzero unless  $f_b(H(u)) = 0$  for every  $b$ , which happens with probability at most  $\delta$  over the choice of  $u \in S$ . ◀

### Indecomposable Polynomials

The *indecomposability* of a polynomial is crucially used in the analysis of the PRG construction in [12] as well as in our analysis. We first define this property.

► **Definition 2.5** (Indecomposability). Let  $f \in \mathbb{F}[\mathbf{x}]$  be a non-constant polynomial over a field  $\mathbb{F}$ . It is said to be *decomposable* over  $\mathbb{F}$  if there exist  $h \in \mathbb{F}[\mathbf{x}]$  and a univariate polynomial  $g \in \mathbb{F}[y]$  such that  $\deg(g) \geq 2$  and  $f = g(h)$ . Otherwise,  $f$  is said to be *indecomposable* over  $\mathbb{F}$ .

Obviously, if a polynomial  $f \in \mathbb{F}_q[\mathbf{x}]$  over a finite field  $\mathbb{F}_q$  is indecomposable over  $\overline{\mathbb{F}}_q$ , then it is also indecomposable over  $\mathbb{F}_q$ . The following lemma, which was proved in [3] and generalized in [5], states that the converse is also true.

► **Lemma 2.6** ([3]; see also [5, Theorem 4.2]). A polynomial  $f \in \mathbb{F}_q[\mathbf{x}]$  that is indecomposable over  $\mathbb{F}_q$  is also indecomposable over  $\overline{\mathbb{F}}_q$ .

In [12], Derksen and Viola proved the following result, which states that if a polynomial is indecomposable, then its outputs are equidistributed.

► **Lemma 2.7** ([12, Lemma 12]). There exists an absolute constant  $C > 0$  such that the following holds: Suppose  $f \in \mathbb{F}_q[\mathbf{x}] = \mathbb{F}_q[x_1, \dots, x_n]$  is indecomposable over  $\mathbb{F}_q$ . Then  $f(\mathbf{U}_{\mathbb{F}_q^n})$  is  $\varepsilon$ -close to  $\mathbf{U}_{\mathbb{F}_q}$ , where  $\varepsilon = Cd^2/\sqrt{q}$ .

The proof of Lemma 2.7 is based on the observation that the indecomposability of  $f$  precisely captures the property that for most  $b \in \mathbb{F}_q$ , the variety  $f^{-1}(b)$ , defined by the constraint  $f(\mathbf{x}) = b$ , is *absolutely irreducible*. This condition of absolute irreducibility is required by the *Weil bound* [36]. Consequently, one can apply the Weil bound to show that for most  $b$ , the number of points in  $f^{-1}(b) \cap \mathbb{F}_q^n$  is close to  $q^{n-1}$ , thereby proving the equidistribution of the output of  $f$ . For details, we refer the reader to [12].

Finally, the following lemma connects indecomposability with irreducibility over algebraically closed fields. It is explicitly stated in, e.g., [9].

► **Lemma 2.8** ([9, Lemma 7]). Let  $f \in \mathbb{F}[\mathbf{x}]$  be a non-constant polynomial over a field  $\mathbb{F}$ . Then  $f$  is indecomposable over  $\overline{\mathbb{F}}$  iff  $f - t$  is irreducible over  $\overline{\mathbb{F}(t)}$ , where  $t$  is a new variable.

**Resultants**

Let  $f(y) = \sum_{i=0}^{d_1} a_i y^i$  and  $g(y) = \sum_{i=0}^{d_2} b_i y^i$  be two univariate polynomials in  $y$  over a field  $\mathbb{F}$  and suppose that  $d_1 + d_2 > 0$ . The *Sylvester Matrix* of  $f$  and  $g$  is the  $(d_1 + d_2) \times (d_1 + d_2)$  matrix

$$\begin{pmatrix} a_0 & & & & b_0 & & & & & \\ a_1 & a_0 & & & b_1 & b_0 & & & & \\ a_2 & a_1 & \ddots & & b_2 & b_1 & \ddots & & & \\ \vdots & & \ddots & a_0 & \vdots & & \ddots & & & b_0 \\ & \vdots & & a_1 & b_{d_2} & \vdots & & & & b_1 \\ a_{d_1} & & & & & b_{d_2} & & & & \\ & a_{d_1} & & \vdots & & & & & & \vdots \\ & & \ddots & & & & \ddots & & & \\ & & & a_{d_1} & & & & & & b_{d_2} \end{pmatrix}.$$

The determinant of this matrix is called the *resultant* of  $f$  and  $g$ , and is denoted  $\text{Res}(f, g)$ . It holds that  $f$  and  $g$  have a common factor if and only if  $\text{Res}(f, g) = 0$  ([11, Proposition 3 in Chapter 3, Section 6]).

Thus, in the case where  $g = \frac{\partial f}{\partial y}$ , it holds that  $\text{Res}(f, g) \neq 0$  if and only if  $f$  does not have a root of multiplicity greater than one.

### Hensel Lifting

Hensel lifting is a general technique for “lifting” roots or factorizations of a polynomial modulo an ideal  $I$  of a ring  $R$  to those modulo powers of  $I$ , under some mild conditions. The use of Hensel’s lifting lemma is standard in multivariate factorization algorithms, and it is available in various forms. We state one standard form, which can be derived from [13, Theorem 7.3] as a special case. This form is particularly relevant to our discussion of Lecerf’s techniques in Section 4.

► **Lemma 2.9** (Hensel’s lifting lemma). Let  $f \in \mathbb{F}[x_1, \dots, x_n, y] = \mathbb{F}[\mathbf{x}, y]$  be a nonzero polynomial over a field  $\mathbb{F}$ . Suppose  $\bar{\lambda} \in \mathbb{F}$  is a simple root of  $f(\mathbf{0}, y) \in \mathbb{F}[y]$ . Then there exists unique  $\lambda \in \mathbb{F}[[\mathbf{x}]]$  such that

1.  $f(\mathbf{x}, \lambda) = 0$ , i.e.,  $\lambda$  is a root of  $f$  as a univariate polynomial in  $y$  over  $\mathbb{F}[[\mathbf{x}]]$ , and
2.  $\lambda(\mathbf{0}) = \bar{\lambda}$ .

## 3 Hypothesis (H)

Lecerf’s papers [20, 21] on multivariate polynomial factoring assume a hypothesis about the polynomial  $f$ , which he calls Hypothesis (H). Such a hypothesis can be satisfied with high probability by applying a random linear transformation on the variables.

In this section, we discuss Lecerf’s Hypothesis (H) and show that, for our purpose, the random linear transformation can be derandomized by using a HSG for polynomials of degree  $O(d)$ . The fact that we are interested in the irreducibility of  $f - t$  for an indeterminate  $t$ , rather than that of  $f$ , is crucial in keeping the degree linear in  $d$ .

Let  $\mathbb{F}$  be a field. First, we define Hypothesis (H).

► **Definition 3.1** (Hypothesis (H) [20, 21]). Let  $f \in \mathbb{F}[x_1, \dots, x_n, y] = \mathbb{F}[\mathbf{x}, y]$  be a non-constant polynomial. We say  $f$  satisfies *Hypothesis (H)* if

1.  $f$  is monic in  $y$  and  $\deg_y(f) = \deg(f)$ ,
2.  $\text{Res}\left(f(\mathbf{0}, y), \frac{\partial f}{\partial y}(\mathbf{0}, y)\right) \neq 0$ .

We also need a family of invertible linear transformations defined as follows.

► **Definition 3.2.** For  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{F}^n$ , let  $s_{\mathbf{a}}$  be the  $\mathbb{F}$ -linear automorphism of  $\mathbb{F}[\mathbf{x}, y]$  that fixes  $y$  and sends  $x_i$  to  $x_i + a_i y$ .

► **Lemma 3.3.** Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be a nonzero polynomial of degree at most  $d$ . Then there exists a nonzero polynomial  $B \in \mathbb{F}[\mathbf{x}]$  of degree at most  $d$  such that for every  $\mathbf{a} \in \mathbb{F}^n$  satisfying  $B(\mathbf{a}) \neq 0$ , it holds that  $\deg_y(s_{\mathbf{a}}(f)) = d$  and the coefficient of  $y^d$  in  $s_{\mathbf{a}}(f)$  is in  $\mathbb{F}^\times$ .

**Proof.** Let  $f_d$  be the degree- $d$  homogeneous part of  $f$ , so that we can write  $f = f_d + g$  where  $g = f - f_d$  has degree less than  $d$ . Write  $f_d = \sum_{i=0}^d c_i(\mathbf{x})y^i$ , where each  $c_i \in \mathbb{F}[\mathbf{x}]$  is either zero or a homogeneous polynomial of degree  $d - i$ .

Consider  $\mathbf{a} \in \mathbb{F}^n$ . Note that

$$s_{\mathbf{a}}(f) = s_{\mathbf{a}}(f_d) + s_{\mathbf{a}}(g) = \sum_{i=0}^d s_{\mathbf{a}}(c_i(\mathbf{x}))y^i + s_{\mathbf{a}}(g) = \sum_{i=0}^d c_i(\mathbf{x} + y \cdot \mathbf{a})y^i + g(\mathbf{x} + y \cdot \mathbf{a}, y).$$

As  $\deg(g) < d$  and each  $c_i$  is either zero or homogeneous of degree  $d - i$ , we have that  $\deg_y s_{\mathbf{a}}(f) \leq d$ , and that the coefficient of  $y^d$  in  $s_{\mathbf{a}}(f)$  is  $\sum_{i=0}^d c_i(\mathbf{a}) \in \mathbb{F}$ . So we may choose  $B = \sum_{i=0}^d c_i$ , which is a nonzero polynomial of degree at most  $d$ .  $\blacktriangleleft$

► **Lemma 3.4.** Assume  $f \in \mathbb{F}[\mathbf{x}, y]$  is a polynomial of degree  $d \geq 1$  that satisfies Item 1 of Hypothesis (H). Further assume that  $\text{char}(\mathbb{F})$  is either zero or greater than  $d$ . Let  $c \in \mathbb{F}^\times$ . Then  $f + ct$  is a degree- $d$  polynomial satisfying Hypothesis (H) as a polynomial over  $\mathbb{F}(t)$ .

**Proof.** As  $f$  satisfies Item 1 of Hypothesis (H) and has degree  $d \geq 1$ , so does  $f + ct$ . So it suffices to verify Item 2. Write  $f = \sum_{i=0}^d c_i y^i$ , where  $c_i \in \mathbb{F}[\mathbf{x}]$  and  $c_d = 1$ . Then  $\frac{\partial(f+ct)}{\partial y} = \sum_{i=1}^d (i \cdot c_i) y^{i-1}$ , which has degree  $d - 1$  in  $y$  since  $d \cdot c_d = d \neq 0$  by the assumption about  $\text{char}(\mathbb{F})$ .

Let  $\bar{c}_i = c_i(\mathbf{0})$  for  $i = 0, 1, \dots, d$ . Let  $h = \text{Res}\left((f + ct)(\mathbf{0}, y), \frac{\partial(f+ct)}{\partial y}(\mathbf{0}, y)\right)$ . Then  $h$  is the determinant of the following  $(2d - 1) \times (2d - 1)$  matrix:

$$\begin{pmatrix} \bar{c}_0 + ct & 0 & \cdots & 0 & \bar{c}_1 & 0 & \cdots & 0 \\ \bar{c}_1 & \bar{c}_0 + ct & \cdots & 0 & 2\bar{c}_2 & \bar{c}_1 & \cdots & 0 \\ \bar{c}_2 & \bar{c}_1 & \ddots & 0 & 3\bar{c}_3 & 2\bar{c}_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & \bar{c}_0 + ct & \vdots & \vdots & \ddots & \bar{c}_1 \\ \bar{c}_d & \bar{c}_{d-1} & \cdots & \vdots & d\bar{c}_d & (d-1)\bar{c}_{d-1} & \cdots & \vdots \\ 0 & \bar{c}_d & \ddots & \vdots & 0 & d\bar{c}_d & \ddots & \vdots \\ \vdots & \vdots & \ddots & \bar{c}_{d-1} & \vdots & \vdots & \ddots & (d-1)\bar{c}_{d-1} \\ 0 & 0 & \cdots & \bar{c}_d & 0 & 0 & \cdots & d\bar{c}_d \end{pmatrix}.$$

Observe that  $\deg_t h \leq d - 1$ , and that the coefficient of  $t^{d-1}$  in  $h$  is  $c^{d-1}(d\bar{c}_d)^d = c^{d-1}d^d \neq 0$  since only those entries on the diagonal contribute to this coefficient. This implies that  $h \neq 0$ , i.e.,  $f + ct$  satisfies Item 2 of Hypothesis (H).  $\blacktriangleleft$

► **Corollary 3.5.** Assume that  $f \in \mathbb{F}[\mathbf{x}, y]$  is a polynomial of degree  $d \geq 1$  and that  $\text{char}(\mathbb{F})$  is either zero or greater than  $d$ . Then there exists a nonzero polynomial  $B \in \mathbb{F}[\mathbf{x}]$  of degree at most  $d$  such that for every  $\mathbf{a} \in \mathbb{F}^n$  satisfying  $B(\mathbf{a}) \neq 0$ ,  $s_{\mathbf{a}}(f) - t$  equals a product  $c \cdot g$  where  $c \in \mathbb{F}^\times$  and  $g \in \mathbb{F}(t)[\mathbf{x}, y]$  is a degree- $d$  polynomial satisfying Hypothesis (H).

**Proof.** Let  $B$  be as in Lemma 3.3. Consider  $\mathbf{a} \in \mathbb{F}^n$  satisfying  $B(\mathbf{a}) \neq 0$ . By Lemma 3.3, we may write  $s_{\mathbf{a}}(f) = c \cdot \tilde{g}$  where  $c \in \mathbb{F}^\times$  and  $\tilde{g}$  satisfies Item 1 of Hypothesis (H). Then  $s_{\mathbf{a}}(f) - t = c \cdot \tilde{g} - t = c \cdot g$  where  $g = \tilde{g} - c^{-1}t$ . By Lemma 3.4,  $g$  is a degree- $d$  polynomial satisfying Hypothesis (H).  $\blacktriangleleft$

Thus, by choosing good  $\mathbf{a} \in \mathbb{F}$  via an explicit HSG for polynomials of degree at most  $d$  and performing the transformation  $f \mapsto s_{\mathbf{a}}(f)$ , we may assume  $f - t$  satisfies Hypothesis (H).

### Satisfying Hypothesis (H) in Small Characteristics

While our final result needs  $\text{char}(\mathbb{F}) > d(d - 1)$ , the assumption that  $\text{char}(\mathbb{F})$  is zero or large enough is not crucial for the sake of satisfying Hypothesis (H). We now sketch how to modify the proof of Lemma 3.4 when  $0 < \text{char}(\mathbb{F}) \leq d$ .

Let  $p = \text{char}(\mathbb{F}) > 0$ . For our purpose, we may assume  $\mathbb{F}$  is a perfect field and  $f$  is indecomposable over  $\mathbb{F}$ . This implies that  $f \notin \mathbb{F}[x_1^p, \dots, x_n^p, y^p]$ . Then it is not hard to show that there exists an integer  $e > 0$  coprime to  $p$  such that for random  $\mathbf{a} \in \mathbb{F}^n$ , with high

## 44:10 Optimal PRGs for Low-Degree Polynomials over Large Fields

probability, not only is the coefficient of  $y^d$  in  $s_{\mathbf{a}}(f)$  nonzero, but so is the coefficient of  $y^e$ . Choose the largest  $e$  that has this property. After replacing  $f$  by  $s_{\mathbf{a}}(f)$ , the polynomial  $\frac{\partial f + ct}{\partial y}$  in the proof of Lemma 3.4 would have degree  $e - 1$  instead of  $d - 1$  in  $y$ . Then  $\deg_t(h) = e - 1$  and the coefficient of  $t^{e-1}$  in  $h$  is  $c^{e-1}(e\bar{c}_e)^d$ , which is nonzero iff  $\bar{c}_e = c_e(\mathbf{0})$  is nonzero. The latter condition can be guaranteed with high probability by performing the substitutions  $x_i \mapsto x_i + b_i$  for random  $\mathbf{b} = (b_1, \dots, b_n) \in \mathbb{F}^n$ . Finally, it is not difficult to show that the choices of  $\mathbf{a}$  and  $\mathbf{b}$  can be derandomized by using an explicit HSG for polynomials of degree  $O(d)$ .

### 4 Lecerf's Techniques

We describe Lecerf's techniques in this section. For simplicity, our discussion is restricted to the special case where the base field is algebraically closed.

Let  $\mathbb{K}$  be an algebraically closed field, and let  $f \in \mathbb{K}[\mathbf{x}, y]$  be a polynomial of degree  $d \geq 1$  satisfying Hypothesis (H). Define  $\bar{f} := f(\mathbf{0}, y) \in \mathbb{K}[y]$ . As  $\mathbb{K}$  is algebraically closed and  $\text{Res}\left(f(\mathbf{0}, y), \frac{\partial f}{\partial y}(\mathbf{0}, y)\right) \neq 0$ , the univariate polynomial  $\bar{f}$  factorizes into distinct linear factors

$$\bar{f} = \prod_{i=1}^d (y - \bar{\lambda}_i)$$

where  $\bar{\lambda}_i \in \mathbb{K}$  for  $i \in [d]$ . By Hensel's lifting lemma, the above factorization of  $\bar{f}$  over  $\mathbb{K}$  lifts to a factorization of  $f$  into distinct linear factors

$$f = \prod_{i=1}^d (y - \lambda_i),$$

where  $\lambda_i \in \mathbb{K}[[\mathbf{x}]]$  and  $\lambda_i(\mathbf{0}) = \bar{\lambda}_i$  for  $i \in [d]$ .

Now we introduce new variables  $\mathbf{z} = (z_1, \dots, z_n)$  and  $x$ , and define  $g := f(z_1x, \dots, z_nx, y) \in \mathbb{K}[\mathbf{z}, x, y]$ . Then  $g$  factorizes into linear factors

$$g = \prod_{i=1}^d (y - \lambda_i(z_1x, \dots, z_nx))$$

where each  $\lambda_i(z_1x, \dots, z_nx)$  lives in  $\mathbb{K}[\mathbf{z}][[x]]$ . For  $i \in [d]$ , let  $g_i$  be the factor  $y - \lambda_i(z_1x, \dots, z_nx)$  of  $g$ , and let  $\hat{g}_i$  be its cofactor  $\prod_{j \in [d] \setminus \{i\}} g_j$ . So  $g_i, \hat{g}_i \in \mathbb{K}[\mathbf{z}][[x]][y]$ .

For  $h \in A[[x]][y]$  over a commutative ring  $A$  and  $(j, k) \in \mathbb{N}^2$ , denote by  $\text{coeff}(h, x^j y^k) \in A$  the coefficient of  $x^j y^k$  in  $h$ . We are now ready to define the linear system  $D_{\mathbf{z}, \sigma}$  used in [20, 21].

► **Definition 4.1** (Linear system  $D_{\mathbf{z}, \sigma}$  [20, 21]). Let  $\sigma \in \mathbb{N}$ . Define  $D_{\mathbf{z}, \sigma}$  to be the following linear system over  $\mathbb{K}(\mathbf{z})$  in the unknowns  $\ell_1, \dots, \ell_d$ :

$$D_{\mathbf{z}, \sigma} \begin{cases} \sum_{i=1}^d \text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial y}, x^j y^k\right) \cdot \ell_i = 0, & k \leq d-1, d \leq j+k \leq \sigma-1, \\ \sum_{i=1}^d \text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial x}, x^j y^k\right) \cdot \ell_i = 0, & k \leq d-1, j \leq \sigma-2, d \leq j+k \leq \sigma-1. \end{cases}$$

We have the following easy lemma.



► **Lemma 4.2.** For  $(j, k) \in \mathbb{N}^2$ ,  $\text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial x}, x^j y^k\right)$ ,  $\text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial y}, x^j y^k\right) \in \mathbb{K}[\mathbf{z}]$  are polynomials of degree at most  $j + 1$  and  $j$  respectively.

**Proof.** Consider arbitrary  $i \in [d]$  and  $(j, k) \in \mathbb{N}^2$ . As  $g_i = y - \lambda_i(z_1 x, \dots, z_n x)$  and  $j \geq 0$ , only terms of degree at most  $j$  in  $z_1, \dots, z_n$  contribute to the coefficient of  $x^j y^k$  in  $g_i$ . Then, as the operator  $\frac{\partial}{\partial x}$  is linear and sends  $x^u y^v$  to  $u x^{u-1} y^v$  for all  $u, v \in \mathbb{N}$ , one can see that only terms of degree at most  $j + 1$  in  $z_1, \dots, z_n$  contribute to the coefficient of  $x^j y^k$  in  $\frac{\partial g_i}{\partial x}$ . Also,  $\frac{\partial g_i}{\partial y} = 1$  by definition.

For any  $h_1, \dots, h_s \in \mathbb{K}[\mathbf{z}][[x]][y]$  and  $h = \prod_{i=1}^s h_i$ , we have

$$\text{coeff}(h, x^j y^k) = \sum_{\substack{j_1, \dots, j_s, k_1, \dots, k_s \in \mathbb{N} \\ \sum_i j_i = j, \sum_i k_i = k}} \prod_{i=1}^s \text{coeff}(h_i, x^{j_i} y^{k_i}). \quad (1)$$

We already know  $\deg(\text{coeff}(g_i, x^j y^k)) \leq j$ ,  $\deg(\text{coeff}(\frac{\partial g_i}{\partial x}, x^j y^k)) \leq j + 1$ , and  $\deg(\text{coeff}(\frac{\partial g_i}{\partial y}, x^j y^k)) = 0$  for  $i \in [d]$  and  $(j, k) \in \mathbb{N}^2$  by the above discussion. Choosing  $(h_1, \dots, h_s)$  to be  $(g_1, \dots, g_{i-1}, g_{i+1}, \dots, g_d, \frac{\partial g_i}{\partial x})$  and  $(g_1, \dots, g_{i-1}, g_{i+1}, \dots, g_d, \frac{\partial g_i}{\partial y})$  respectively and applying (1) proves the claim. ◀

For  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{K}^n$ , we can assign  $a_1, \dots, a_n$  to  $z_1, \dots, z_n$  respectively in the polynomials  $\text{coeff}(\hat{g}_i \frac{\partial g_i}{\partial x}, x^j y^k)$ ,  $\text{coeff}(\hat{g}_i \frac{\partial g_i}{\partial y}, x^j y^k) \in \mathbb{K}[\mathbf{z}]$ . This yields a linear system over  $\mathbb{K}$ , called the *specialization* of  $D_{\mathbf{z}, \sigma}$  at  $\mathbf{a}$  and denoted by  $D_{\mathbf{a}, \sigma}$ .

► **Definition 4.3 (Specialization).** For  $\sigma \in \mathbb{N}$  and  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{K}^n$ , define  $D_{\mathbf{a}, \sigma}$  to be the following linear system over  $\mathbb{K}$  in the unknowns  $\ell_1, \dots, \ell_d$ :

$$D_{\mathbf{a}, \sigma} \begin{cases} \sum_{i=1}^d \text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial y}, x^j y^k\right)(\mathbf{a}) \cdot \ell_i = 0, & k \leq d - 1, d \leq j + k \leq \sigma - 1, \\ \sum_{i=1}^d \text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial x}, x^j y^k\right)(\mathbf{a}) \cdot \ell_i = 0, & k \leq d - 1, j \leq \sigma - 2, d \leq j + k \leq \sigma - 1. \end{cases}$$

For  $S \subseteq [d]$ , define  $\delta_S = (\delta_{S,1}, \dots, \delta_{S,d}) \in \mathbb{K}^d$  by

$$\delta_{S,i} = \begin{cases} 1 & i \in S, \\ 0 & i \notin S. \end{cases}$$

For every factor  $\tilde{f}$  of  $f$ , we may associate a set  $S \subseteq [d]$  such that  $\tilde{f} = \prod_{i \in S} (y - \lambda_i)$ , i.e.,  $S$  is the set of indices  $i \in [d]$  such that  $y - \lambda_i$  divides  $\tilde{f}$ . The irreducible factors  $f_1, \dots, f_r$  of  $f$  over  $\mathbb{K}$  are then associated with sets  $S_1, \dots, S_r \subseteq [d]$ , which form a partition of  $[d]$ . In [20, 21], Lecerf proved that, when  $\sigma$  is large enough, the solution space of  $D_{\mathbf{z}, \sigma}$  is exactly spanned by the vectors  $\delta_{S_1}, \dots, \delta_{S_r}$ , and a similar statement holds for the specializations  $D_{\mathbf{a}, \sigma}$ . We state Lecerf's results formally as the following theorem.

► **Theorem 4.4** ([20, 21]). Assume  $\text{char}(\mathbb{K})$  is zero or greater than  $d(d - 1)$ . Let  $\sigma \geq 2d$ . Let  $f \in \mathbb{K}[\mathbf{x}, y]$  be a polynomial of degree  $d \geq 1$  satisfying Hypothesis (H). Then:

1. Suppose  $f = \prod_{i=1}^r f_i$  is the factorization of  $f$  into its irreducible factors over  $\mathbb{K}$ . For  $i \in [r]$ , let  $S_i$  be the set of indices  $j \in [d]$  such that  $y - \lambda_j$  divides  $f_i$ . Then  $\delta_{S_1}, \dots, \delta_{S_r}$  form a basis of the solution space of  $D_{\mathbf{z}, \sigma}$ .

## 44:12 Optimal PRGs for Low-Degree Polynomials over Large Fields

2. Let  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{K}^n$  and  $f_{\mathbf{a}} = f(a_1x, \dots, a_nx, y) \in \mathbb{K}[x, y]$ . Suppose  $f_{\mathbf{a}} = \prod_{i=1}^s f_{\mathbf{a},i}$  is the factorization of  $f_{\mathbf{a}}$  into its irreducible factors over  $\mathbb{K}$ . For  $i \in [s]$ , let  $S_{\mathbf{a},i}$  be the set of indices  $j \in [d]$  such that  $y - \lambda_j(a_1x, \dots, a_nx)$  divides  $f_{\mathbf{a},i}$ . Then  $\delta_{S_{\mathbf{a},1}}, \dots, \delta_{S_{\mathbf{a},s}}$  form a basis of the solution space of  $D_{\mathbf{a},\sigma}$ .

The first item of Theorem 4.4 is explicitly stated as [21, Lemma 1]. The second item follows from [20, Theorem 1 and Lemma 4]. For a detailed analysis of the linear systems  $D_{\mathbf{z},\sigma}$  and  $D_{\mathbf{a},\sigma}$ , and for a conceptual interpretation of these linear systems in terms of the closedness condition of differential 1-forms, we refer the reader to [20, 21], as well as to the earlier paper of Gao [15].

### Bertinian Good/Bad Points

The classical Bertini irreducibility theorem [30] states, among other things, that over an algebraically closed field  $\mathbb{K}$ , the intersection of an irreducible affine variety of codimension one (i.e., a hypersurface) with a plane in general position is still irreducible. This motivates the following definition:

► **Definition 4.5** (Bertinian good/bad points [21]). Let  $f \in \mathbb{K}[\mathbf{x}, y]$  be a non-constant polynomial satisfying Hypothesis (H). We say  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{K}^n$  is a *Bertinian good point* for  $f$  if for every irreducible factor  $\tilde{f}$  of  $f$  over  $\mathbb{K}$ , the bivariate polynomial  $\tilde{f}_{\mathbf{a}} = \tilde{f}(a_1x, \dots, a_nx, y)$  is also irreducible over  $\mathbb{K}$ . We say  $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{K}^n$  is a *Bertinian bad point* for  $f$  if it is not a Bertinian good point for  $f$ .

Lecerf [21, Theorem 6] proved that given  $f$ , there exists a nonzero polynomial  $Q \in \mathbb{K}[z_1, \dots, z_n]$  of degree at most  $(d-1)(2d-1)$  that vanishes at all Bertinian bad points for  $f$ , where  $d = \deg(f)$ . Let  $M$  be the matrix representing the linear system  $D_{\mathbf{z},\sigma}$ . Lecerf's proof can be sketched as follows: By Theorem 4.4, the solution space of  $D_{\mathbf{a},\sigma}$  contains that of  $D_{\mathbf{z},\sigma}$ , and  $\mathbf{a}$  is Bertinian good as long as the two are equal. Thus, we may choose  $Q$  to be the determinant of the largest nonsingular submatrix of  $M$ . This is because for such  $Q$ , if  $Q$  does not vanish at  $\mathbf{a}$ , then  $D_{\mathbf{z},\sigma}$  and  $D_{\mathbf{a},\sigma}$  have the same rank, and hence their solution spaces must be equal. The bound  $(d-1)(2d-1)$  on the degree of  $Q$  follows from Lemma 4.2.

In [21], Lecerf also demonstrated that the degree bound  $(d-1)(2d-1)$  is asymptotically tight by providing an example for which a degree of  $\Omega(d^2)$  of the polynomial  $Q$  is necessary. However, our next lemma states that, perhaps surprisingly, the degree bound can be improved to  $2d-1$  if we allow the use of the zero loci of *multiple* polynomials to cover the Bertinian bad points for  $f$ . For simplicity, we state the lemma in the special case where  $f$  is irreducible, which suffices for our purpose.

► **Lemma 4.6.** Assume  $\text{char}(\mathbb{K})$  is zero or greater than  $d(d-1)$ . Let  $f \in \mathbb{K}[\mathbf{x}, y]$  be an irreducible polynomial over  $\mathbb{K}$  of degree  $d \geq 1$  satisfying Hypothesis (H). Let  $m = 2^{d-1} - 1$ . Then there exist nonzero polynomials  $Q_1, \dots, Q_m \in \mathbb{K}[\mathbf{z}] = \mathbb{K}[z_1, \dots, z_n]$  of degree at most  $2d-1$  such that for every Bertinian bad point  $\mathbf{a} \in \mathbb{K}^n$  for  $f$ , at least one polynomial  $Q_i$  vanishes at  $\mathbf{a}$ .

**Proof.** Let  $\sigma = 2d$ . Let  $N$  be the number of equations in  $D_{\mathbf{z},\sigma}$ . Let  $M$  be the  $N \times d$  matrix over  $\mathbb{F}(\mathbf{z})$  representing the linear system  $D_{\mathbf{z},\sigma}$ . Note that by Definition 4.1, the entries of  $M$  are of the form  $\text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial x}, x^j y^k\right)$  with  $j \leq \sigma - 2$  or  $\text{coeff}\left(\hat{g}_i \frac{\partial g_i}{\partial y}, x^j y^k\right)$  with  $j \leq \sigma - 1$ . By Lemma 4.2 and the fact that  $\sigma = 2d$ , the entries of  $M$  are polynomials in  $\mathbb{K}[\mathbf{z}]$  of degree at most  $2d-1$ .

There are exactly  $m = 2^{d-1} - 1$  proper subsets of  $[d]$  containing 1. Let  $S_1, \dots, S_m$  be an enumeration of them. Consider  $i \in [m]$ . As  $f$  is irreducible, by Theorem 4.4,  $\delta_{S_i}$  is not in the solution space of  $D_{\mathbf{z}, \sigma}$ . So we can fix a row  $\mathbf{r}_i = (r_{i,1}, \dots, r_{i,d})$  of  $M$  such that the inner product of  $\mathbf{r}_i$  and  $\delta_{S_i}$  is nonzero, i.e.,  $\sum_{j=1}^d r_{i,j} \delta_{S_i,j} \neq 0$ . Let  $Q_i = \sum_{j=1}^d r_{i,j} \delta_{S_i,j}$ , which is a nonzero polynomial in  $\mathbb{K}[\mathbf{z}]$  of degree at most  $2d - 1$ . Choose  $Q_i$  in this way for each  $i = 1, \dots, m$ .

Now let  $\mathbf{a}$  be a Bertinian bad point for  $f$ . Then  $f_{\mathbf{a}} = f(a_1x, \dots, a_nx, y)$  factorizes into more than one irreducible factor over  $\mathbb{K}$ . Let  $\tilde{f}_{\mathbf{a}}$  be the irreducible factor of  $f_{\mathbf{a}}$  divisible by  $y - \lambda_1(a_1x, \dots, a_nx)$ . Let  $S$  be the set of  $j \in [d]$  such that  $\tilde{f}_{\mathbf{a}}$  is divisible by  $y - \lambda_j(a_1x, \dots, a_nx)$ . Then  $S$  is a proper subset of  $[d]$  containing 1. So  $S = S_i$  for some  $i \in [m]$ . By Theorem 4.4,  $\delta_{S_i}$  is in the solution space of  $D_{\mathbf{a}, \sigma}$ . As  $D_{\mathbf{a}, \sigma}$  is the specialization of  $D_{\mathbf{z}, \sigma}$  at  $\mathbf{a}$ , the vector  $(r_{i,1}(\mathbf{a}), \dots, r_{i,d}(\mathbf{a}))$  is a row of the matrix representing  $D_{\mathbf{a}, \sigma}$ . So  $\sum_{j=1}^d r_{i,j}(\mathbf{a}) \delta_{S_i,j} = 0$ , i.e.,  $Q_i(\mathbf{a}) = 0$ . ◀

### The Number of Low-Degree Polynomials Needed

It is an intriguing mathematical question to us how many low-degree polynomials are needed to cover the Bertinian bad points for  $f$ . We now formalize this question.

▶ **Definition 4.7.** Let  $\mathbb{K}$  be an algebraically closed field. For positive integers  $d$  and  $D$ , define  $N(d, D, \mathbb{K})$  to be the smallest  $N \in \mathbb{N}$  such that the following holds: Let  $f \in \mathbb{K}[\mathbf{x}, y]$  be an irreducible polynomial of degree at most  $d$  over  $\mathbb{K}$  satisfying Hypothesis (H). Then there exist  $N$  nonzero polynomials in  $\mathbb{K}[\mathbf{z}]$  of degree at most  $D$  such that the union of the zero loci of these polynomials contains all Bertinian bad points for  $f$  in  $\mathbb{K}^n$ .

If such  $N$  does not exist, define  $N(d, D, \mathbb{K}) = \infty$ .

In our application, it suffices to consider polynomials of the special form  $f + c \cdot t$ , where  $c \in \mathbb{F}^\times$ ,  $f \in \mathbb{F}[\mathbf{x}, y]$  and  $\mathbb{K} = \overline{\mathbb{F}(t)}$ . Moreover, by performing a variable substitution  $t \mapsto -c^{-1}t$ , we may assume  $c = -1$ . This motivates us to introduce the following variant of Definition 4.7:

▶ **Definition 4.8.** Let  $\mathbb{F}$  be a field. For positive integers  $d$  and  $D$ , define  $N^*(d, D, \mathbb{F})$  to be the smallest  $N \in \mathbb{N}$  such that the following holds: Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be a polynomial of degree at most  $d$  such that  $f - t$  is an irreducible polynomial over  $\overline{\mathbb{F}(t)}$  satisfying Hypothesis (H). Then there exist  $N$  nonzero polynomials in  $\overline{\mathbb{F}(t)}[\mathbf{z}]$  of degree at most  $D$  such that the union of the zero loci of these polynomials contains all Bertinian bad points for  $f - t$  in  $\overline{\mathbb{F}^n}$ .

If such  $N$  does not exist, define  $N^*(d, D, \mathbb{F}) = \infty$ .

Lecerf's result [21, Theorem 6] can be interpreted as the statement that when  $\text{char}(\mathbb{K})$  is zero or greater than  $d(d - 1)$ , it holds that

$$N(d, D, \mathbb{K}) = 1 \text{ for } D \geq (d - 1)(2d - 1).$$

Our Lemma 4.6 states that under the same condition, we have

$$N(d, D, \mathbb{K}) \leq 2^{d-1} - 1 \text{ for } D \geq 2d - 1.$$

In [21], Lecerf gave an example showing that the degree bound  $O(d^2)$  for the smallest  $D$  satisfying  $N(d, D, \mathbb{K}) = 1$  is asymptotically tight.<sup>2</sup> As one can always combine the  $N(d, D, \mathbb{K})$  polynomials of degree at most  $D$  into a single polynomial of degree at most  $N(d, D, \mathbb{K}) \cdot D$  by taking their product, this implies  $N(d, D, \mathbb{K}) \cdot D = \Omega(d^2)$ , i.e.,  $N(d, D, \mathbb{K}) = \Omega(d^2/D)$ .

<sup>2</sup> See the example before Theorem 6 in [21].

► **Question 4.9.** Give improved upper bounds (or lower bounds) on  $N(d, D, \mathbb{K})$  and  $N^*(d, D, \mathbb{F})$ , at least when the characteristic of  $\mathbb{K}$  or  $\mathbb{F}$  is zero or large enough.

By definition,  $N^*(d, D, \mathbb{F}) \leq N(d, D, \overline{\mathbb{F}(t)})$ . A subexponential upper bound on  $N^*(d, D, \mathbb{F})$  for  $D = O(d)$  will improve the required field size in Theorem 1.1.

Finally, it might be possible to exploit some extra structure to derive better bounds on  $N^*(d, D, \mathbb{F})$  than those obtained for  $N(d, D, \mathbb{K})$ . For example, if we modify the definition of  $N^*(d, D, \mathbb{F})$  by only considering those polynomials  $f$  of *prime* degree, then  $N^*(d, 2d-1, \mathbb{F}) \leq 1$ . This is because if  $f_{\mathbf{a}} - t$  is reducible over  $\overline{\mathbb{F}(t)}$ , then  $f_{\mathbf{a}}$  is decomposable over  $\overline{\mathbb{F}}$  by Lemma 2.8. But as  $\deg(f)$  is prime,  $f_{\mathbf{a}}$  must be of the form  $g(h)$  with  $\deg(g) = \deg(f)$  and  $\deg(h) = 1$ . This in turn implies that  $f_{\mathbf{a}} - t$  factorizes into  $\deg(f)$  linear factors over  $\overline{\mathbb{F}(t)}$ . In Theorem 5.4, we use this idea to show that the required field size can be improved to  $O(d^4/\varepsilon^2)$  if we only want to fool polynomials whose degrees are prime and at most  $d$ .

## 5 Proofs of the Main Theorems

In this section, we present our PRG construction and prove the main theorems.

Let  $n$  and  $d$  be positive integers. Let  $\mathbb{F}_q$  be a finite field of characteristic at least  $d(d-1)+1$ . We now present the construction of our PRG

$$G : S \rightarrow \mathbb{F}_q^{n+1}$$

for polynomials  $f \in \mathbb{F}_q[\mathbf{x}, y] = \mathbb{F}_q[x_1, \dots, x_n, y]$  of degree at most  $d$ . To simplify our notation, these polynomials are assumed to be  $(n+1)$ -variate rather than  $n$ -variate.

► **Construction 5.1.** The construction is as follows:

- Let  $H : T \rightarrow \mathbb{F}_q^n$  be an explicit HSG for  $n$ -variate polynomials of degree at most  $2d-1$  over  $\mathbb{F}_q$  with density  $1-\delta$  and seed length  $\log |T| = O(d \log n + \log(1/\delta))$ , where  $\delta = C_0(2d-1)/q$  and  $C_0 > 0$  is an absolute constant. For  $i \in [n]$  and  $s \in T$ , denote the  $i$ -th coordinate of  $H(s)$  by  $H(s)_i$ . The existence of  $H$  is guaranteed by Theorem 2.3.
- Let  $S = T \times T \times \mathbb{F}_q \times \mathbb{F}_q$ . Define  $G : S \rightarrow \mathbb{F}_q^{n+1}$  by

$$G(r, s, u, v) = (H(s)_1 \cdot u + H(r)_1 \cdot v, \dots, H(s)_n \cdot u + H(r)_n \cdot v, v).$$

In other words, we use random  $(r, s) \in T \times T$  to pick a plane in  $\mathbb{F}_q^{n+1}$ , and use random  $(u, v) \in \mathbb{F}_q \times \mathbb{F}_q$  to pick a point on the plane. The following lemma states that with high probability, a given indecomposable polynomial  $f \in \mathbb{F}_q[\mathbf{x}, y]$  remains indecomposable when restricted to the plane.

► **Lemma 5.2.** Let  $f \in \mathbb{F}_q[\mathbf{x}, y]$  be an indecomposable polynomial of degree at most  $d$  over  $\mathbb{F}_q$ . Let  $(r, s)$  be a random element of  $T \times T$ . Let  $\mathbf{a} = (a_1, \dots, a_n) = H(r)$  and  $\mathbf{b} = (b_1, \dots, b_n) = H(s)$ . Finally, let  $F = f(b_1x + a_1y, \dots, b_nx + a_ny, y) \in \mathbb{F}_q[x, y]$ . Then

$$\Pr[F \text{ is indecomposable over } \mathbb{F}_q] \geq 1 - 2^{d-1}\delta.$$

**Proof.** Recall that  $s_{\mathbf{a}}$  is the  $\mathbb{F}_q$ -linear automorphism of  $\mathbb{F}_q[\mathbf{x}, y]$  that fixes  $y$  and sends  $x_i$  to  $x_i + a_iy$ . As  $f$  is indecomposable over  $\mathbb{F}_q$ , so is  $s_{\mathbf{a}}(f)$ . By Lemma 2.6,  $s_{\mathbf{a}}(f)$  is also indecomposable over  $\overline{\mathbb{F}_q}$ . So  $s_{\mathbf{a}}(f) - t$  is irreducible over  $\overline{\mathbb{F}_q(t)}$  by Lemma 2.8.

By Corollary 3.5, there exists a nonzero polynomial  $B \in \mathbb{F}_q[\mathbf{x}]$  of degree at most  $d$  such that if  $B(\mathbf{a}) \neq 0$ , then

$$s_{\mathbf{a}}(f) - t = c \cdot g \tag{2}$$

where  $c \in \mathbb{F}_q^\times$  and  $g \in \mathbb{F}_q(t)[\mathbf{x}, y] \subseteq \overline{\mathbb{F}_q(t)}[\mathbf{x}, y]$  is a degree- $d$  polynomial satisfying Hypothesis (H). By the HSG property of  $H$ , the event  $B(\mathbf{a}) \neq 0$  happens with probability at least  $1 - \delta$ . Condition on this event, so that (2) holds. As  $s_{\mathbf{a}}(f) - t$  is irreducible over  $\overline{\mathbb{F}_q(t)}$ , so is  $g$ .

Let  $m = 2^{d-1} - 1$ . By Lemma 4.6, there exist nonzero polynomials  $Q_1, \dots, Q_m \in \overline{\mathbb{F}_q(t)}[z_1, \dots, z_n]$  of degree at most  $2d - 1$  such that the union of the zero loci of these polynomials contains all  $\mathbf{b}^* = (b_1^*, \dots, b_n^*) \in \overline{\mathbb{F}_q}^n$  for which  $g(b_1^*x, \dots, b_n^*x, y)$  is reducible over  $\overline{\mathbb{F}_q(t)}$ . By Fact 2.4,  $H$  is an HSG with density  $1 - \delta$  for polynomials of degree at most  $2d - 1$  over  $\overline{\mathbb{F}_q(t)}$ .<sup>3</sup> Therefore, for each  $i \in [m]$ , the probability that  $Q_i(\mathbf{b}) = 0$  is at most  $\delta$ . Condition on the event  $Q_1(\mathbf{b}), \dots, Q_m(\mathbf{b}) \neq 0$ . Then  $g(b_1x, \dots, b_nx, y)$  is irreducible over  $\overline{\mathbb{F}_q(t)}$ . On the other hand, note that

$$c \cdot g(b_1x, \dots, b_nx, y) \stackrel{(2)}{=} (s_{\mathbf{a}}(f))(b_1x, \dots, b_nx, y) - t = f(b_1x + a_1y, \dots, b_nx + a_ny, y) - t = F - t$$

where the second step uses the definition  $s_{\mathbf{a}}(f) = f(x_1 + a_1y, \dots, x_n + a_ny, y) \in \mathbb{F}_q[\mathbf{x}, y]$ . So  $F - t$  is irreducible over  $\overline{\mathbb{F}_q(t)}$ . By Lemma 2.8,  $F$  is indecomposable over  $\overline{\mathbb{F}_q}$ . So it is indecomposable over  $\mathbb{F}_q$ .

The indecomposability of  $F$  over  $\mathbb{F}_q$  relies on the conditions  $B(\mathbf{a}) \neq 0$  and  $Q_1(\mathbf{b}), \dots, Q_m(\mathbf{b}) \neq 0$ . By the union bound, these conditions are simultaneously satisfied with probability at least  $1 - \delta - m\delta = 1 - 2^{d-1}\delta$ , which completes the proof. ◀

Now we are ready to prove Theorem 1.1.

► **Theorem 5.3** (Theorem 1.1 restated). There exists an absolute constant  $C > 0$  such that for  $\varepsilon > 0$  and  $q \geq C(d2^d/\varepsilon + d^4/\varepsilon^2)$  with  $\text{char}(\mathbb{F}_q) \geq d(d-1) + 1$ ,  $G$  as in Construction 5.1 is a PRG for  $(n+1)$ -variate polynomials of degree at most  $d$  over  $\mathbb{F}_q$  with error  $\varepsilon$  and seed length  $O(d \log n + \log q)$ .

**Proof.** Let  $f \in \mathbb{F}_q[\mathbf{x}, y]$  be a polynomial of degree at most  $d$ . We want to prove that  $f(G(\mathbf{U}_S))$  and  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}})$  are  $\varepsilon$ -close (in statistical distance). We may assume that  $f$  is a non-constant polynomial, i.e.,  $\deg(f) \geq 1$ , since the claim is trivial otherwise.

Our next step is the same as in [12]:  $f$  can always be written in the form  $f = g(h)$ , where  $g \in \mathbb{F}_q[z]$  is a univariate polynomial and  $h \in \mathbb{F}_q[\mathbf{x}, y]$  is indecomposable over  $\mathbb{F}_q$ . Let  $D = h(G(\mathbf{U}_S))$  and  $D' = h(\mathbf{U}_{\mathbb{F}_q^{n+1}})$ . Then  $f(G(\mathbf{U}_S)) = g(D)$  and  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}}) = g(D')$ . If  $D$  and  $D'$  are  $\varepsilon$ -close, then  $g(D)$  and  $g(D')$  are also  $\varepsilon$ -close. Thus, by replacing  $f$  with  $h$ , we may assume that  $f$  is indecomposable over  $\mathbb{F}_q$ .

Let  $r, s, \mathbf{a}, \mathbf{b}$  and  $F$  be as in Lemma 5.2. Then by Lemma 5.2, the probability that  $F$  is decomposable over  $\mathbb{F}_q$  over random  $r$  and  $s$  is at most  $2^{d-1}\delta = C_0 2^{d-1}(2d-1)/q$ , where  $C_0$  is as in Construction 5.1. Fix  $r$  and  $s$  such that  $F$  is indecomposable over  $\mathbb{F}_q$ . Then  $f(G(r, s, u, v)) = F(u, v)$  by definition. Applying Lemma 2.7 to  $F$  shows that, for such fixed  $r$  and  $s$ , the distribution of  $F(u, v)$ , i.e.,  $f(G(r, s, u, v))$ , over random  $u, v \in \mathbb{F}_q$  is  $\varepsilon'$ -close to  $\mathbf{U}_{\mathbb{F}_q}$ , where  $\varepsilon' = C_1 d^2/\sqrt{q}$  and  $C_1 > 0$  is an absolute constant. It follows that the statistical distance between  $f(G(\mathbf{U}_S))$  and  $\mathbf{U}_{\mathbb{F}_q}$  is at most  $2^{d-1}\delta + \varepsilon'$ .

On the other hand, as  $f$  is also indecomposable over  $\mathbb{F}_q$ , applying Lemma 2.7 to  $f$  shows that  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}})$  is  $\varepsilon'$ -close to  $\mathbf{U}_{\mathbb{F}_q}$ . Therefore, the statistical distance between  $f(G(\mathbf{U}_S))$  and  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}})$  is at most

$$(2^{d-1}\delta + \varepsilon') + \varepsilon' = 2^{d-1}\delta + 2\varepsilon' = C_0 2^{d-1}(2d-1)/q + 2C_1 d^2/\sqrt{q} \tag{3}$$

<sup>3</sup> Note that we are applying Fact 2.4 to the infinite extension  $\overline{\mathbb{F}_q(t)}/\mathbb{F}_q$ . In principle, it should be possible to make the argument finitary by making some adaptations, such as considering specific values of  $t$ . However, this may increase the complexity of the proof.

## 44:16 Optimal PRGs for Low-Degree Polynomials over Large Fields

which is bounded by  $\varepsilon$  provided that  $q \geq C(d2^d/\varepsilon + d^4/\varepsilon^2)$  and  $C > 0$  is a large enough absolute constant. The seed length of  $G$  is

$$2 \log |T| + 2 \log q = O(d \log n + \log(1/\delta) + \log q) = O(d \log n + \log q)$$

as  $\delta = C_0(2d - 1)/q$ . ◀

We conclude this section by proving Theorem 1.2, which states that the required field size can be improved to  $O(d^4/\varepsilon^2)$  if we only want to fool polynomials of prime degree.

► **Theorem 5.4** (Theorem 1.2 restated). There exists an absolute constant  $C > 0$  such that for  $\varepsilon > 0$  and  $q \geq C(d^4/\varepsilon^2)$  with  $\text{char}(\mathbb{F}_q) \geq d(d - 1) + 1$ ,  $G$  as in Construction 5.1 is a PRG for  $(n + 1)$ -variate polynomials of *prime* degree up to  $d$  with error  $\varepsilon$  and seed length  $O(d \log n + \log q)$ .

**Proof Sketch.** Let  $f \in \mathbb{F}_q[\mathbf{x}, y]$  be a polynomial whose degree  $d'$  is prime and at most  $d$ . We want to prove that  $f(G(\mathbf{U}_S))$  and  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}})$  are  $\varepsilon$ -close (in statistical distance). Suppose  $f$  is decomposable over  $\mathbb{F}_q$ . Then  $f = g(h)$  for some polynomials  $g, h$  over  $\mathbb{F}_q$  where  $\deg(g) \geq 2$ , and as  $d' = \deg(f)$  is prime, we must have  $\deg(g) = d'$  and  $\deg(h) = 1$ . In this case, the theorem follows by replacing  $f$  with  $h$ , which has degree one, and applying Theorem 5.3. So we may assume that  $f$  is indecomposable over  $\mathbb{F}_q$ .

The rest of the proof follows that of Theorem 5.3, except that we could bound the probability that  $F$  is decomposable over  $\mathbb{F}_q$  by  $2\delta$ , rather than by  $2^{d-1}\delta$ , using the following observation:

In the application of Lemma 4.6, the polynomial has the special form  $g^* = f^* + ct$ , where  $f^* \in \mathbb{F}_q[\mathbf{x}, y]$ ,  $c \in \mathbb{F}_q^\times$ , and  $\deg(f^*) = d'$ . By making the substitution  $t \mapsto -c^{-1}t$ , we may assume  $c = -1$ . Consider any  $\mathbf{a} \in \overline{\mathbb{F}_q}^n$  such that  $g_{\mathbf{a}}^* = g^*(a_1x, \dots, a_nx, y)$  is reducible over  $\overline{\mathbb{F}_q}(t)^n$ . We claim that  $g_{\mathbf{a}}^*$  factorizes into linear factors over  $\overline{\mathbb{F}_q}(t)$ . To see this, note that  $f_{\mathbf{a}}^* = f^*(a_1x, \dots, a_nx, y)$  is a decomposable polynomial over  $\overline{\mathbb{F}_q}$  of degree  $d'$  by Lemma 2.8 and the fact that  $g_{\mathbf{a}}^* = f_{\mathbf{a}}^* - t$  is reducible over  $\overline{\mathbb{F}_q}(t)$ . So we may write  $f_{\mathbf{a}}^* = \alpha(\beta)$  where  $\alpha \in \overline{\mathbb{F}_q}[z]$ ,  $\beta \in \overline{\mathbb{F}_q}[\mathbf{x}, y]$ , and  $\deg(\alpha) > 1$ . As  $d'$  is prime, we must have  $\deg(\alpha) = d'$  and  $\deg(\beta) = 1$ . As  $\alpha$  is univariate,  $\alpha - t$  factorizes into linear factors  $\alpha_1, \dots, \alpha_{d'}$  over  $\overline{\mathbb{F}_q}(t)$ . So  $g_{\mathbf{a}}^* = \alpha(\beta) - t = (\alpha - t)(\beta)$  factorizes into the linear factors  $\alpha_1(\beta), \dots, \alpha_{d'}(\beta)$  over  $\overline{\mathbb{F}_q}(t)$ .

This observation shows that there is only one bad factorization pattern to rule out, namely, the complete factorization into linear factors. This allows us to save a factor of  $2^{d-1} - 1$  and reduce the error probability in Lemma 5.2 from  $(2^d - 1)\delta + \delta$  to  $\delta + \delta = 2\delta$ . The bound on the statistical distance between  $f(G(\mathbf{U}_S))$  and  $f(\mathbf{U}_{\mathbb{F}_q^{n+1}})$  in (3) now becomes  $2\delta + 2\varepsilon' = 2C_0(2d - 1)/q + 2C_1d^2/\sqrt{q}$ , which is bounded by  $\varepsilon$  provided that  $q \geq C(d^4/\varepsilon^2)$  and  $C > 0$  is a large enough absolute constant. ◀

## 6 Open Problems

We conclude with some open problems. The most obvious one is reducing the required field size in our construction. Using Bogdanov's [6] paradigm, it seems necessary for the field to be of size at least polynomial in  $d$ , since this argument relies on the Weil bound (and indeed, as mentioned in Section 1, the seed lengths of the known constructions over small fields like  $\mathbb{F}_2$  are worse). Still, one could hope to obtain seed length  $O(d \log n)$  with  $q$  being polynomial in  $d$ , and not exponential in  $d$ . In our construction,  $q$  is exponential in  $d$  due to the need to apply a union bound over all possible vectors in  $\{0, 1\}^d$  characterizing the factorization pattern of  $f_{\mathbf{a}}$ . It could very well be that there is a more clever argument that rules out

multiple vectors at once. We also mention again Question 4.9. As explained in Section 4, improved upper bounds on the quantity  $N^*(d, D, \mathbb{F})$  in that question would improve the field size required by our construction.

A related open problem is removing the requirement that the characteristic of  $\mathbb{F}_q$  is at least  $d(d-1)+1$ . This requirement comes from using Lecerf’s [21] arguments (dating back to Gao [15] and Ruppert [26, 27]). We remark that our construction can be adapted to work in arbitrary characteristics  $p > 0$  by employing the analysis in [8] and increasing the accuracy parameter  $\sigma$  of Hensel lifting to  $d(d-1)+1$ ; however, this leads to a larger seed length of  $O(d^2 \log n + \log q)$ . A proof of this result will be provided in the full version of this paper.

Finally, low-degree polynomials form a natural “weak” class of polynomials. However, rather than assuming bounds on the degree of polynomials, one can also consider other weak classes of polynomials, where the restriction comes from bounding their algebraic circuit complexity. This forms another interesting avenue for generalizing the results on PRGs for low-degree polynomials. As an analogy, in the context of Boolean computation, the problem of constructing explicit PRGs for weak computational classes (such as bounded-depth circuits or read-once oblivious branching programs) is well studied (see [34]). For algebraic computational models, however, much less is known. Most of the research in this area has focused on constructing *hitting sets* of limited models of algebraic circuits (see [31, 28, 29] for some surveys on this topic), due to the relation to the famous Polynomial Identity Testing Problem. To the best of our knowledge, there is no known explicit construction of PRGs for any natural class of algebraic computation. A concrete and intriguing open problem is to explicitly construct PRGs for the class of sparse polynomials, for which, as described in the references above, there are many known explicit constructions of hitting sets.

---

## References

- 1 Noga Alon, Ido Ben-Eliezer, and Michael Krivelevich. Small sample spaces cannot fool low degree polynomials. In *Proceedings of the 12th International Workshop on Randomization and Computation (RANDOM 2008)*, volume 5171 of *Lecture Notes in Computer Science*, pages 266–275. Springer, 2008. doi:10.1007/978-3-540-85363-3\_22.
- 2 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992. doi:10.1002/RSA.3240030308.
- 3 I. V. Arzhantsev and A. P. Petravchuk. Closed polynomials and saturated subalgebras of polynomial algebras. *Ukrainian Mathematical Journal*, 59(12):1783–1790, 2007.
- 4 Avraham Ben-Aroya and Amnon Ta-Shma. Constructing small-bias sets from algebraic-geometric codes. *Theory of Computing*, 9:253–272, 2013. doi:10.4086/TOC.2013.V009A005.
- 5 Arnaud Bodin, Pierre Debes, and Salah Najib. Indecomposable polynomials and their spectrum. *Acta Arithmetica*, 139(1):79–100, 2009.
- 6 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 21–30. ACM, 2005. doi:10.1145/1060590.1060594.
- 7 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM Journal on Computing*, 39(6):2464–2486, 2010. doi:10.1137/070712109.
- 8 A. Bostan, G. Lecerf, B. Salvy, É. Schost, and B. Wiebelt. Complexity issues in bivariate polynomial factorization. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, ISSAC 2004*, pages 42–49. Association for Computing Machinery, 2004.
- 9 Guillaume Cheze and Salah Najib. Indecomposability of polynomials via Jacobian matrix. *Journal of Algebra*, 324(1):1–11, 2010.

- 10 Gil Cohen and Amnon Ta-Shma. Pseudorandom generators for low degree polynomials from algebraic geometry codes. *Electronic Colloquium on Computational Complexity*, TR13-155, 2013. [arXiv:TR13-155](#).
- 11 David A. Cox, John B. Little, and Donal O’Shea. *Ideals, Varieties and Algorithms*. Undergraduate Texts in Mathematics. Springer, 2007. [doi:10.1007/978-0-387-35651-8](#).
- 12 Harm Derksen and Emanuele Viola. Fooling polynomials using invariant theory. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 399–406. IEEE, 2022. [doi:10.1109/FOCS54457.2022.00045](#).
- 13 David Eisenbud. *Commutative Algebra: With a View Toward Algebraic Geometry*. Springer Science & Business Media, 1995.
- 14 Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Efficient approximation of product distributions. *Random Structures & Algorithms*, 13(1):1–16, 1998. [doi:10.1002/\(SICI\)1098-2418\(199808\)13:1<1::AID-RSA1>3.0.CO;2-W](#).
- 15 Shuhong Gao. Factoring multivariate polynomials via partial differential equations. *Mathematics of Computation*, 72(242):801–822, 2003. [doi:10.1090/S0025-5718-02-01428-X](#).
- 16 Venkatesan Guruswami and Chaoping Xing. Hitting sets for low-degree polynomials with optimal density. In *Proceedings of the IEEE 29th Conference on Computational Complexity, CCC 2014*, pages 161–168. IEEE Computer Society, 2014. [doi:10.1109/CCC.2014.24](#).
- 17 Pooya Hatami and William Hoza. Theory of unconditional pseudorandom generators. *Electronic Colloquium on Computational Complexity*, TR23-019, 2023. [arXiv:TR23-019](#).
- 18 Erich L. Kaltofen. Effective Noether irreducibility forms and applications. *Journal of Computer and System Sciences*, 50(2):274–295, 1995. [doi:10.1006/JCSS.1995.1023](#).
- 19 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 (STOC 2001)*, pages 216–223. ACM, 2001. [doi:10.1145/380752.380801](#).
- 20 Grégoire Lecerf. Sharp precision in Hensel lifting for bivariate polynomial factorization. *Mathematics of Computation*, 75(254):921–933, 2006.
- 21 Grégoire Lecerf. Improved dense multivariate polynomial factorization algorithms. *Journal of Symbolic Computation*, 42(4):477–494, 2007.
- 22 Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. *Theory of Computing*, 5(1):69–82, 2009. [doi:10.4086/TOC.2009.V005A003](#).
- 23 Chi-Jen Lu. Hitting set generators for sparse polynomials over any finite fields. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012*, pages 280–286. IEEE Computer Society, 2012. [doi:10.1109/CCC.2012.20](#).
- 24 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993. [doi:10.1137/0222053](#).
- 25 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Matematicheskie Zametki*, 41:598–607, 1987. English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, 41(4):333-338, 1987. [doi:10.1007/BF01137685](#).
- 26 Wolfgang Ruppert. Reduzibilität ebener kurven. *Journal für die reine und angewandte Mathematik*, 1986(369):167–191, 1986. [doi:doi:10.1515/crll.1986.369.167](#).
- 27 Wolfgang M. Ruppert. Reducibility of polynomials  $f(x, y)$  modulo  $p$ . *Journal of Number Theory*, 77(1):62–70, 1999. [doi:10.1006/jnth.1999.2381](#).
- 28 Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.
- 29 Nitin Saxena. Progress on polynomial identity testing-II. *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, pages 131–146, 2014. [doi:10.1007/978-3-319-05446-9\\_7](#).
- 30 Igor Shafarevich. *Basic Algebraic Geometry 1: Varieties in Projective Space*. Springer, 1994.




- 31 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/04000000039.
- 32 Roman Smolensky. On representations by low-degree polynomials. In *34th Annual Symposium on Foundations of Computer Science*, pages 130–138. IEEE Computer Society, 1993. doi:10.1109/SFCS.1993.366874.
- 33 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. ACM, 2017. doi:10.1145/3055399.3055408.
- 34 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/04000000010.
- 35 Emanuele Viola. The sum of  $D$  small-bias generators fools polynomials of degree  $D$ . *computational complexity*, 18(2):209–217, 2009. doi:10.1007/S00037-009-0273-5.
- 36 André Weil. Numbers of solutions of equations in finite fields. *Bulletin of the American Mathematical Society*, 55:497–508, 1949.



# Refining the Adaptivity Notion in the Huge Object Model

Tomer Adar  

Technion – Israel Institute of Technology, Haifa, Israel

Eldar Fischer 

Technion – Israel Institute of Technology, Haifa, Israel

---

## Abstract

The Huge Object model for distribution testing, first defined by Goldreich and Ron in 2022, combines the features of classical string testing and distribution testing. In this model we are given access to independent samples from an unknown distribution  $P$  over the set of strings  $\{0, 1\}^n$ , but are only allowed to query a few bits from the samples. The distinction between adaptive and non-adaptive algorithms, which occurs naturally in the realm of string testing (while being irrelevant for classical distribution testing), plays a substantial role also in the Huge Object model.

In this work we show that the full picture in the Huge Object model is much richer than just that of the adaptive vs. non-adaptive dichotomy. We define and investigate several models of adaptivity that lie between the fully-adaptive and the completely non-adaptive extremes. These models are naturally grounded by observing the querying process from each sample independently, and considering the “algorithmic flow” between them. For example, if we allow no information at all to cross over between samples (up to the final decision), then we obtain the *locally bounded* adaptive model, arguably the “least adaptive” one apart from being completely non-adaptive. A slightly stronger model allows only a “one-way” information flow. Even stronger (but still far from being fully adaptive) models follow by taking inspiration from the setting of streaming algorithms. To show that we indeed have a hierarchy, we prove a chain of exponential separations encompassing most of the models that we define.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Huge-Object model, Property Testing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.45

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2306.16129>

**Funding** *Eldar Fischer*: Research supported by an Israel Science Foundation grant number 879/22.

## 1 Introduction

Property testing is the study of sublinear, query-based probabilistic decision-making algorithms. That is, algorithms that return ACCEPT or REJECT after reading only a small portion of their input. The study of (classical) property testing, starting with [6], [14] and [15], has seen an extensive body of work. See for example [10]. Usually, a property-testing algorithm with threshold parameter  $\varepsilon$  is required to accept an input that satisfies the property with high probability, and reject an input whose distance from any satisfying one is more than  $\varepsilon$ , with high probability as well. For string properties, which were the first to be studied (along with functions, matrices, etc. that can also be represented as strings), the distance measure is usually the normalized Hamming distance.

Distribution testing is a newer model, first defined implicitly in [11] (a version of which has already appeared in 2000 as a technical report). In [4] and [5] it was explicitly defined and researched. The algorithms in this model are much weaker, where instead of queries, the



© Tomer Adar and Eldar Fischer;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 45; pp. 45:1–45:16



Leibniz International Proceedings in Informatics

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

decision to accept or reject must be made based only on a sequence of independent samples drawn from an unknown distribution. In such a setting the distance metric is usually the variation distance. For a more comprehensive survey, see [7].

The study of a combination of string and distribution testing was initiated in [12]. Here the samples in themselves are considered to be very large objects, and hence after obtaining a sample (usually modeled as a string of size  $n$ ), queries must be made to obtain some information about its contents. This requires an appropriate modification in the distance notion. This model is appropriately called the *Huge Object model*.

Contrast the above to the original “small object” distribution testing model, where it is assumed that every sample is immediately available to the algorithm in its entirety. In particular, in the original model, the algorithm does not have any choice of queries, as it just receives a sequence of independent samples from the distribution to be tested. Hence one might even call it a “formula” rather than an “algorithm”. Grossly speaking, the only decision made is whether to accept or reject the provided sequence of sampled objects.

On the other hand, in the string testing model, an algorithm is provided with a (deterministic) input string, and may make query decisions based both on internal random coins and on answers to previous queries. An algorithm which makes use of the option of considering answers to previous queries when choosing the next query is called adaptive, while an algorithm that queries based only on coin tosses is called non-adaptive (the final decision on whether to accept or reject the input must, of course, depend on the actual answers).

Algorithms for the Huge Object model, due to their reliance on individual queries to the provided samples, can be adaptive or non-adaptive. This relationship with respect to the Huge Object model was first explored in [8].

However, as we shall demonstrate below, the complete picture here is richer than the standard adaptive/non-adaptive dichotomy used in classical string testing. As it turns out, several categories of adaptivity can be defined and investigated based on the consideration of the shared information between the different samples that are queried.

## 1.1 Adaptivity notions in the Huge Object model

For our purpose, unless we state otherwise, we assume that the sequence of samples is taken in advance (but is not directly disclosed to the algorithm), and is presented as a matrix from which the algorithm makes its queries. For a sequence of  $s$  samples from a distribution whose base set is  $\{0, 1\}^n$ , this would be a binary  $s \times n$  matrix.

We say that an algorithm is *non-adaptive* if it chooses its entire set of queries before making them, which means that it cannot choose later queries based on the answers to earlier ones. This is identical to the definition of a non-adaptive algorithm for string properties.

A *fully adaptive* algorithm is allowed to choose every query based on answers to all queries made before it. This is quite similar to the definition of an adaptive algorithm for string properties, but restricting ourselves to this dichotomy does not give the full picture. We refine the notion of adaptivity by considering more subtle restrictions on the way that the algorithms plan their queries, leading to query models that are not as expressive as those of fully adaptive algorithms, but are still more expressive than those of non-adaptive ones. In this introduction we only introduce the rationale of every model; the formal definitions appear in the preliminaries section.

One interesting restriction, which is surprisingly difficult to analyze, is “being adaptive for every individual sample, without sharing adaptivity between different samples” (the results of random coin tosses are still allowed to be shared). We say that an algorithm is

*locally-bounded* if it obeys this restriction. This model captures the concept of distributed execution, in a way that every node has a limited scope of a single sample, and only when all nodes are done, their individual outcomes are combined to facilitate a decision.

A more natural restriction is “being able to query only the most recent sample”. We say that an algorithm is *forward-only* if it cannot query a sample after querying a later one. This can be viewed (if we abandon the above-mentioned matrix representation) as the algorithm being provided with oracle access to only one sample at a time, not being able to “go back in time” once a new sample was taken. An example for the usage of the model is an anonymous survey. As long as the survey session is alive, we can present new questions based on past interactions and on the current one, but once the session ends, we are not able to recall the same participant for further questioning.

A natural generalization of forward-only adaptiveness is having a bounded memory for holding samples (rather than only having one accessible sample at a time). Once the memory is full, the algorithm must drop one of these samples (making it inaccessible) in order to free up space for a new sample. An additional motivation for this model is the concept of stream processing, whose goal is computing using sublinear memory. Relevant to our work is [2], where the input stream is determined by an unknown distribution, in contrast to the usual streaming setting where the order of the stream is arbitrary. Within the notion of having memory of a fixed size, we actually distinguish two models. In the weak model, when the memory is full, the oldest sample is dropped. In the strong model, the algorithm decides (possibly adaptively) which sample to drop.

We show that every two consecutive models in the above hierarchy have an exponential separation, which means that there is a property that requires  $\Omega(\text{poly}(n))$  queries for an  $\varepsilon$ -test in the first model (for some fixed  $\varepsilon$ ), but is also  $\varepsilon$ -testable using  $O(\text{poly}(\varepsilon^{-1}) \log n)$  queries in the second model (for every  $\varepsilon > 0$ ). Moreover, our upper bounds always have one-sided error, while the lower bounds apply for both one-sided and two-sided error algorithms. The exact relationship between the weak and the strong limited memory models remains open, however.

We believe that investigating limited adaptiveness models can apply to other areas where there are two “query scales”. That is, when investigating a model takes into account collections of objects that are restricted both in the way that whole objects are obtained and in the access model *inside* each obtained object. For example, one could think of a distributed computing scenario where the communication between the nodes follows a LOCAL or a CONGEST scheme (see [13]), but additionally each node holds a “large” input from which it may only perform sub-linear time computation *between* the communication rounds.

## 1.2 Organization of the paper

We start with formal definitions of the models which are required to state our results, followed by an overview of the results themselves and a description of the main ideas of their proofs. This review includes the definitions of the properties showing our separation results, along with a sketch of the lower bounds and the algorithms for the upper bounds. The proofs themselves are deferred to the full version of this paper.

## 2 Preliminaries

The following are the core definitions and lemmas used throughout this paper, including the model definitions used in the overview in Section 3. Here, all distributions are defined over finite sets.

► **Definition 1** (Common notations). For a set  $A$ , the power set of  $A$  is denoted by  $\mathcal{P}(A)$ . For two sets  $A$  and  $B$ , the set of all functions  $f : A \rightarrow B$  is denoted by  $B^A$ . For a finite set  $A$ , the set of all permutations over  $A$  is denoted by  $\pi(A)$ .

► **Definition 2** (Set of distributions). Let  $\Omega$  be a finite set. The set of all distributions that are defined over  $\Omega$  is denoted by  $\mathcal{D}(\Omega)$ .

While parts of this section are generalizable to distributions over non-finite sets  $\Omega$  with compact topologies, we restrict ourselves to distributions over finite sets, which suffice for our application.

► **Definition 3** (Property). A property  $\mathcal{P}$  over a finite alphabet  $\Sigma$  is defined as a sequence of compact sets  $\mathcal{P}_n \subseteq \mathcal{D}(\Sigma^n)$ . Here compactness refers to the one defined with respect to the natural topology inherited from  $\mathbb{R}^{|\Sigma|^n}$ .

All properties are defined over  $\Sigma = \{0, 1\}$  unless we state otherwise.

## 2.1 Distances

The following are the distance measures that we use. In the sequel, we will omit the subscript (e.g. use “ $d(x, y)$ ” instead of “ $d_H(x, y)$ ”) whenever the measure that we use is clear from the context.

► **Definition 4** (Normalized Hamming distance). For two strings  $s_1, s_2 \in \Sigma^n$ , we use  $d_H(s_1, s_2)$  to denote their normalized Hamming distance,  $\frac{1}{n} |\{1 \leq i \leq n \mid s_1[i] \neq s_2[i]\}|$ .

For all our distance measures we also use the standard extension to distances between sets, using the corresponding infimum (which in all our relevant cases will be a minimum). For example, For a string  $s \in \{0, 1\}^n$  and a set  $A \subseteq \{0, 1\}^n$ , we define  $d_H(s, A) = \min_{s' \in A} d_H(s, s')$ .

► **Definition 5** (Variation distance). For two distributions  $P$  and  $Q$  over a common set  $\Omega$ , we use  $d_{\text{var}}(P, Q)$  to denote their variation distance,  $\max_{E \subseteq \Omega} |\Pr_P[E] - \Pr_Q[E]|$ . Since  $\Omega$  is finite there is an equivalent definition of  $d_{\text{var}}(P, Q) = \frac{1}{2} \sum_{s \in \Omega} |P(s) - Q(s)|$ .

► **Definition 6** (Transfer distribution). For two distributions  $P$  over  $\Omega_1$  and  $Q$  over  $\Omega_2$ , we say that a distribution  $T$  over  $\Omega_1 \times \Omega_2$  is a transfer distribution between  $P$  and  $Q$  if for every  $x_0 \in \Omega_1$ ,  $\Pr_{(x, y) \sim T}[x = x_0] = \Pr_P[x_0]$ , and for every  $y_0 \in \Omega_2$ ,  $\Pr_{(x, y) \sim T}[y = y_0] = \Pr_Q[y_0]$ . We use  $\mathcal{T}(P, Q)$  to denote the set of all transfer distributions between  $P$  and  $Q$ .

We note that for finite  $\Omega_1$  and  $\Omega_2$  the set  $\mathcal{T}(P, Q)$  is compact as a subset of  $\mathcal{D}(\Omega_1 \times \Omega_2)$ .

► **Definition 7** (Earth Mover’s Distance). For two distributions  $P$  and  $Q$  over a common set  $\Omega$  with a metric  $d_\Omega$ , we use  $d_{\text{EMD}}(P, Q)$  to denote their earth mover’s distance, defined by the infimum of the “average distance” demonstrated by a transfer distribution,  $\inf_{T \in \mathcal{T}(P, Q)} \mathbb{E}_{(x, y) \sim T} [d_\Omega(x, y)]$ .

In the sequel, the above “inf” can and will be replaced by “min”, by the compactness of  $\mathcal{T}(P, Q)$  for finite  $\Omega$ . Most papers (including the original [12]) use an equivalent definition that is based on linear programming, whose solution is the optimal transfer distribution.

In our theorems,  $\Omega$  is always  $\{0, 1\}^n$  for some  $n$  and the metric is the Hamming distance. Sometimes, as an intermediate phase, we may use a different  $\Omega$  (usually  $\{1, \dots, k\}^n$  for some  $k$ ), and then show a reduction back to the binary case.

► **Definition 8** (Distance from a property). *The distance of a distribution  $P$  from a property  $\mathcal{P} = \langle \mathcal{P}_n \rangle$  is loosely noted as  $d(P, \mathcal{P})$  and is defined to be  $d_{\text{EMD}}(P, \mathcal{P}_n) = \inf_{Q \in \mathcal{P}_n} d_{\text{EMD}}(P, Q)$ .*

It is very easy to show that for any two distributions  $P, Q \in \mathcal{D}(\Sigma^n)$  we have  $d_{\text{EMD}}(P, Q) \leq d_{\text{var}}(P, Q)$ . This means that the topology induced by the variation distance is richer than that induced by the earth mover's distance (actually for finite sets these two topologies are identical). In particular it means that all considered properties form compact sets with respect to the earth mover's distance. We obtain the following lemma.

► **Lemma 9.** *For a property  $\mathcal{P}$  of distributions over strings, and any distribution  $P \in \mathcal{D}(\Sigma^n)$ , there is a distribution realizing the distance of  $P$  from  $\mathcal{P}$ , i.e. a distribution  $Q \in \mathcal{P}_n$  for which  $d(P, Q) = d(P, \mathcal{P}_n)$ . In particular, the infimum in Definition 8 is a minimum.*

## 2.2 The testing model

This model is defined in [12]. We use an equivalent definition which will be the “baseline” for our restricted adaptivity variants.

The input is a distribution  $P$  over  $\Sigma^n$  (our final theorems will be for  $\Sigma = \{0, 1\}$ , but some intermediate arguments require other finite  $\Sigma$ ). An algorithm  $\mathcal{A}$  gets random oracle access to  $s$  samples that are independently drawn from  $P$ . Then it is allowed to query individual bits of the samples. The output of the algorithm is either ACCEPT or REJECT. For convenience we identify the samples with an  $s \times n$  matrix, so for example the query “ $(i, j)$ ” returns the  $j$ th bit of the  $i$ th sample.

The input size  $n$  and the number of samples  $s$  are hard-coded in the algorithm. As with boolean circuits, an algorithm for an arbitrarily sized input is defined as a sequence of algorithms, one for each  $n$ .

For a given algorithm we define another measure of complexity, which is the total number of queries that the algorithm makes. Without loss of generality, we always assume that every sample is queried at least once (implying that  $q \geq s$ ).

For a property  $\mathcal{P}$  and  $\varepsilon > 0$ , we say that an algorithm  $\mathcal{A}$  is an  $\varepsilon$ -test if:

- For every  $P \in \mathcal{P}$ ,  $\mathcal{A}$  accepts the input  $P$  with probability higher than  $\frac{2}{3}$ .
- For every  $P$  that is  $\varepsilon$ -far from  $\mathcal{P}$ ,  $\mathcal{A}$  accepts the input  $P$  with probability less than  $\frac{1}{3}$ .

We say that  $\mathcal{A}$  is an  $\varepsilon$ -test with one sided error if:

- For every  $P \in \mathcal{P}$ ,  $\mathcal{A}$  accepts the input  $P$  with probability 1.
- For every  $P$  that is  $\varepsilon$ -far from  $\mathcal{P}$ ,  $\mathcal{A}$  accepts the input  $P$  with probability less than  $\frac{1}{2}$ .

The choice of the probability bounds in the above definition are somewhat arbitrary. For the one sided error definition  $\frac{1}{2}$  is more convenient than  $\frac{1}{3}$ . We also note that for non- $\varepsilon$ -far inputs that are not in  $\mathcal{P}$ , any answer by  $\mathcal{A}$  is considered to be correct.

## 2.3 Restricted models

As observed by Yao in [16], every probabilistic algorithm can be seen as a distribution over the set of allowable deterministic algorithms. This simplifies the algorithmic analysis, since we only have to consider deterministic algorithms (a distinction between public and private coins can break this picture, but this will not be the case here). We will use Yao's observation to define every probabilistic algorithmic model by defining its respective set of allowable deterministic algorithms. The following definitions formalize the description of the models introduced in Section 1.

► **Definition 10** (Fully adaptive algorithm). *Every deterministic algorithm can be described as a full decision tree  $T$  and a set  $A$  of accepted leaves. Without loss of generality we assume that all leaves have the exactly the same depth (we use dummy queries if “padding” is needed). Every internal node of  $T$  consists of a query  $(i, j) \in \{1, \dots, s\} \times \{1, \dots, n\}$  (the  $j$ th bit of the  $i$ th sample), and every edge corresponds to an outcome element (in  $\Sigma$ ). The number of queries  $q$  is defined as the height of the tree. Every leaf can be described by the string of length  $q$  detailing the answers given to the  $q$  queries, corresponding to its root-to-leaf path. Thus we can also identify  $A$  with a subset of  $\Sigma^q$ .*

Now that we have defined the most general form of a deterministic algorithm in the Huge Object model, we formally define our models for varying degrees of adaptivity.

► **Definition 11** (Non-adaptive algorithm). *We say that an algorithm is non-adaptive if it chooses its queries in advance, rather than deciding each query location based on the answers to its previous ones. Formally, every deterministic non-adaptive algorithm is described as a pair  $(Q, A)$  such that  $Q \subseteq \{1, \dots, s\} \times \{1, \dots, n\}$  (for some sample complexity  $s$ ) is the set of queries, and  $A \subseteq \Sigma^Q$  is the set of accepted answer functions. The query complexity is defined as  $q = |Q|$ .*

► **Definition 12** (Locally-bounded adaptive algorithm). *We call an algorithm locally-bounded if it does not choose its queries to a sample based on answers to queries in other samples. Formally, every  $s$ -sample deterministic locally-bounded algorithm is a tuple  $(T_1, \dots, T_s; A)$ , where every  $T_i$  is a decision tree of height  $q_i$  (where  $q = \sum_{i=1}^s q_i$  is the total number of queries) that is only allowed to query the  $i$ th sample, and  $A \subseteq \Sigma^q$  represents a set of accepted superleaves, where a superleaf is defined as the concatenation of the  $q_1, \dots, q_s$  symbol long sequences that represent the leaves of trees  $T_1, \dots, T_s$  respectively.*

► **Definition 13** (Forward-only adaptive algorithm). *We call an algorithm forward-only if it cannot query a sample after querying a later one. Formally, a forward-only algorithm for  $s$  samples of  $n$ -length strings is defined as a pair  $(T, A)$ , where  $T$  is a decision tree over  $\{1, \dots, s\} \times \{1, \dots, n\}$  and  $A \subseteq \Sigma^q$  (as with general adaptive algorithms), additionally satisfying that for every internal node of  $T$  that is not the root, if its query is  $(i, j)$  and its parent query is  $(i', j')$ , then  $i' \leq i$ .*

► **Definition 14** (Weak memory-bounded adaptive algorithm). *We say that an algorithm is weak  $m$ -memory bounded if it can only query a sliding window of the  $m$  most recent samples at a time. Formally, a weak  $m$ -memory-bounded adaptive algorithm using  $s$  samples of  $n$ -length strings is defined as a pair  $(T, A)$ , where  $T$  is a decision tree over  $\{1, \dots, s\} \times \{1, \dots, n\}$  and  $A \subseteq \Sigma^q$  (as with general adaptive algorithms), additionally satisfying that for every internal node of  $T$  that is not the root, if its query is  $(i, j)$ , then for every ancestor whose query is  $(i', j')$ , it holds that  $i' - m < i$ .*

► **Definition 15** (Strong memory-bounded adaptive algorithm). *A strong memory-bounded adaptive algorithm for  $s$  samples of  $n$ -length strings is defined as a triplet  $(T, A, M)$  where  $T$  is a decision tree,  $A \subseteq \Sigma^q$  is the set of accepted answer vectors, and  $M : \text{nodes}(T) \rightarrow \mathcal{P}(\{1, \dots, s\})$  is the “memory state” at every node. The explicit rules of  $M$  are:*

- For every internal node  $u \in T$ ,  $|M(u)| \leq k$  (there are at most  $k$  samples in memory).
- For every internal node  $u \in T$ , if  $i \in M(u)$ , and if  $v$  is a child of  $u$  for which  $i \notin M(v)$ , then for every descendant  $w$  of  $v$ ,  $i \notin M(w)$  (a “forgotten” sample cannot be “recalled”).
- For every internal node  $u \in T$  whose query is  $(i, j)$ ,  $i \in M(u)$  (the  $i$ th sample must be in memory in order to query it).



Without loss of generality, because the samples are independent, we can assume that:

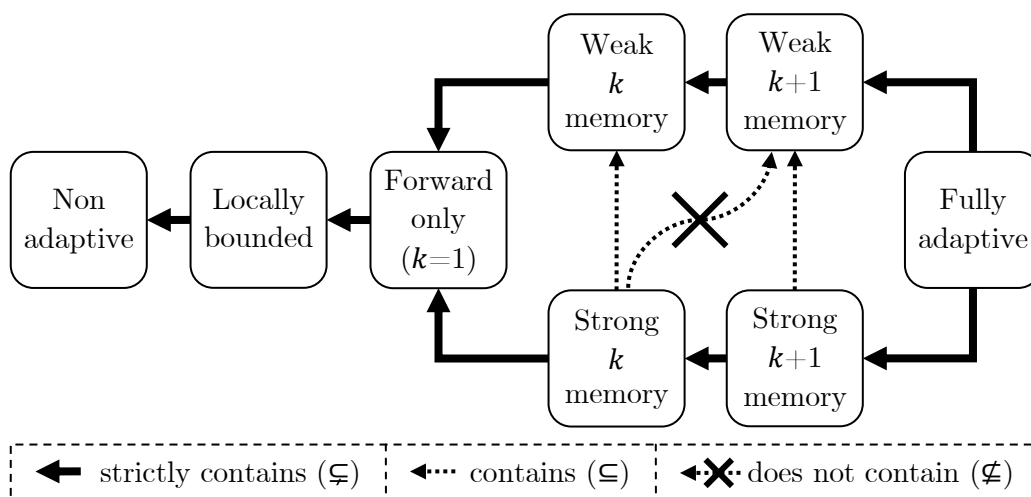
- $M(\text{root}) = \{1, \dots, k\}$  (the algorithm has initial access to the first  $k$  samples).
- For every internal node  $u \in T$  and the set  $V$  of all its ancestors, it holds that  $\max(M(u)) \leq 1 + \max_{v \in V}(\max M(v))$  (new samples are accessed “in order”).

### 3 Overview of results and methods

The following is a semi-formal overview of our work, which is described in extensive details in the full version of the paper. Most of our results are exponential separations between models (that is,  $O(\log n)$  vs  $n^{\Omega(1)}$  bounds).

All separations are with an exponential gap, and are achieved by properties that have an efficient 1-sided error test in one model, but do not even have an efficient 2-sided test in the other model.

Figure 1 provides a visualization of our results. More details about the difference between the weak  $k$ -memory and the strong  $k$ -memory model are provided below.



■ **Figure 1** Graphical summary of our results.

#### 3.1 Non-adaptive algorithms

To showcase what can be done with non-adaptive algorithms, we analyze the property of a distribution having support size at most  $m$ , and the even more basic property of a distribution being *deterministic*, that is, having support size 1.

We show that the determinism property (the property that the distribution draws a specific element with probability 1) can be tested non-adaptively using  $O(\varepsilon^{-1})$  queries, consisting of  $O(\varepsilon^{-1})$  samples (as in the classic model) and  $O(1)$  queries per sample.

► **Observation 16.** *The property of drawing a fixed string has a one-sided error non-adaptive  $\varepsilon$ -test that uses  $O(\varepsilon^{-1})$  queries.*

We also show a non-adaptive  $m$ -support test.

► **Theorem 17.** *The property of being supported on a set of at most  $m$  elements has a one-sided error non-adaptive  $\varepsilon$ -test that uses  $O(\varepsilon^{-2}m \log m)$  queries.*

Algorithm 1 demonstrates this upper bound.

■ **Algorithm 1** One sided  $\varepsilon$ -test for  $m$ -bounded support, non adaptive,  $O(\varepsilon^{-2}m \log m)$  queries.

---

```

take  $s = 1 + \lceil 8\varepsilon^{-1}m \rceil$  samples.
let  $t = \lceil 4\varepsilon^{-1}(\ln m + 2) \rceil$ 
choose  $j_1, \dots, j_t \in [n]$  uniformly and independently at random.
let  $J = \{j_1, \dots, j_t\}$ 
for  $i$  from 1 to  $s$  do
    query sample  $i$  at  $j$  for every  $j \in J$ , giving substring  $y^i$  of length  $|J|$ .
if  $|\{y^1, \dots, y^s\}| > m$  then
    return REJECT
return ACCEPT

```

---

As described in detail in the full version of the paper, in which we prove the correctness of Algorithm 1, our  $\varepsilon$ -test for the  $m$ -support property needs more than a fixed number of queries per sample. Though not necessarily optimal, this algorithm demonstrates the core difference between the Huge Object model and the classic one: the limited ability to distinguish different samples. This limitation holds for adaptive algorithms as well, even though the adaptivity can reduce the number of queries per sample for some properties. A concurrent work [1] shows a lower bound of  $\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$  queries for non-adaptive support testing even for  $m = 2$ , showing that this limitation is unavoidable.

## Locally bounded adaptive algorithms

The locally-bounded adaptive model (Definition 12) allows the algorithm to pick its queries based on answers to previous queries for every *fixed* sample, but lacks the ability to pass information between samples. The ability of being adaptive allows the algorithm more ways to query its samples, but it still lacks the ability to test *relations* between the samples.

### Analysis method

To analyze the locally-bounded model, we define an intermediate model of string testing which we call the *split-adaptive model*.

► **Definition 18** (Split adaptive algorithm). *For a fixed  $k$ , a  $k$ -split adaptive deterministic algorithm for  $n$ -long strings (where  $n$  is divisible by  $k$ ) over some alphabet  $\Sigma$  is a sequence of  $k$  decision trees  $T_1, \dots, T_k$ , where the tree  $T_i$  can only query at indexes between  $(i-1)k + 1$  and  $ik$ , and a set of accepted answer sequences. The query complexity of the algorithm is defined as the sum of heights of its trees.*

In this model, we test properties of  $k$ -tuples of strings, where the queries are made separately for every entry of the tuple (that is, every entry is processed using an adaptive algorithm that is oblivious of the other entries). To obtain a reduction, we consider every  $s$ -sample locally-bounded algorithm over an input distribution  $P$  as a split-adaptive algorithm whose input is drawn from  $P^s$  (that is, an  $s$ -tuple whose entries are independently drawn from  $P$ ).

### Exponential separation from the non-adaptive model

Naturally, there is an exponential separation between the locally-bounded model and the non-adaptive model of the Huge Object model. The property **CPal** (defined below) demonstrates this separation.

► **Definition 19** (string property **cpal**, see [8], [3]). For any fixed  $n$ , the property **cpal** is defined over  $\{0, 1, 2, 3\}^n$  as the set of  $n$ -long strings that are concatenations of a palindrome over  $\{0, 1\}$  and a palindrome over  $\{2, 3\}$  (in this order).

The following lemma is well-known (the adaptive bound, using binary search, is described in [8]).

► **Lemma 20.** Property **cpal** does not have a non-adaptive  $\frac{1}{5}$ -test using  $o(\sqrt{n})$  queries, while having an adaptive  $\varepsilon$ -test using  $O(\log(n) + 1/\varepsilon)$  many queries.

In [8] this was made into a distribution property by using “distributions” that are deterministic.

► **Definition 21** (Distribution property **CPal**, see [8]). For a fixed, even  $n$ , the property **CPal** is defined as the set of distributions over  $\{0, 1\}^n$  that are deterministic (have support size 1), whose support is an element that belongs to **cpal**, with respect to the encoding  $(0, 1, 2, 3) \mapsto (00, 01, 10, 11)$ .

► **Lemma 22.** **CPal** has a locally-bounded  $\varepsilon$ -test that uses  $O(\text{poly}(\varepsilon^{-1}) \log n)$  queries for every  $\varepsilon > 0$ , but there exists some  $\varepsilon_0 > 0$  for which any non-adaptive  $\varepsilon_0$ -test requires  $\Omega(\text{poly}(n))$  queries.

This is an almost-direct corollary of a result from [12] regarding converting string testing problems to the Huge Object model. Essentially, the Huge Object model “contains” the string testing one, and the conversion produces locally adaptive algorithms out of their respective adaptive string algorithms.

## Forward only adaptive algorithms

In the forward-only model (Definition 13), the algorithm virtually gets a stream of samples, and is allowed to query only the current sample without any restriction (but further queries to past samples are not allowed), based on answers to all past queries. In contrast to the locally bounded model, forward algorithms can test a richer collection of binary relations between samples, due to the ability to query one sample and then use the gathered data to choose the queries for the next one.

## Exponential separation from the locally-bounded model

We use the ability of forward-only algorithms to consider a richer collection of relations between samples, as compared to locally-bounded algorithms, to show an exponential separation between these models. The property **Inv\*** (defined below) demonstrates this separation.

In [9] it was shown that  $\varepsilon$ -testing two functions over  $\{1, \dots, n\}$  for being inverses of each other is possible with  $O(\varepsilon^{-1})$  many queries, while testing a single function for having an inverse is harder and requires a polynomial number of queries. Formally, we cite the function property **inv**:

► **Definition 23** (Function property **inv**). For a fixed  $n$ , the property **inv** is defined over  $[n]^{[2n]}$  as the set of ordered pairs of functions  $f, g : [n] \rightarrow [n]$  such that either  $f(i) = g(i)$  for every  $1 \leq i \leq n$  or  $g(f(i)) = i$  for every  $1 \leq i \leq n$ .

## 45:10 Refining the Adaptivity Notion in the Huge Object Model

Note that we modified the definition of the property slightly from the original, by allowing also the case  $f = g$ . This technical change makes it possible to construct a test using forward-only adaptivity that is also with one-sided error.

Here we separate the two functions by setting them in a probability space with support size 2. If we allow forward-only adaptivity, then the original inverse test can be implemented, as it works by verifying that  $g(f(i)) = i$  for sufficiently many  $i$ s. We can call the first sample “ $f$ ”, and after writing down our  $f(i_1), \dots, f(i_q)$ , we “wait” for a sample of  $g$  and then verify that  $g(f(i_j)) = i_j$  for  $i_1, \dots, i_q$ . Formally, we define the property **Inv**:

► **Definition 24** (Distribution property **Inv**). *For a fixed  $n$ , the property **Inv** is defined as the set of distributions over  $[n]^{[n]}$  that are supported by a set of the form  $\{f, g\}$  such that  $(f, g) \in \mathbf{inv}$ . Note that in particular all deterministic distributions satisfy **Inv**, since we allow  $f = g$  to occur.*

To make the above work for binary strings (rather than an alphabet of size  $n$ ) we use an appropriate large distance encoding of the values.

► **Definition 25** (Distribution property **Inv**<sup>\*</sup>). *For a fixed  $n$ , let  $C_n : [n] \rightarrow \{0, 1\}^{2^{\lceil \log_2 n \rceil}}$  be an error-correction code whose distance is at least  $\frac{1}{3}$ . We define **Inv**<sup>\*</sup> as the property of distributions over  $\{0, 1\}^{2^{\lceil \log_2 n \rceil} n}$  that can be constructed by the following procedure: beginning with some  $P \in \mathbf{Inv}$ , we let  $P^*$  denote the distribution that draws  $x \in [n]^n$  according to  $P$ , and then outputs the concatenation  $C_n(x_1) \cdots C_n(x_n)$ .*

The lower bound against locally-bounded adaptivity requires an intricate analysis of the model. Essentially we use the split-adaptive string-testing model to show that when querying each of  $f$  and  $g$  “in solitude”, being adaptive over a function that is drawn at random does not provide an advantage over a non-adaptive algorithm. In particular, the values of a uniformly drawn permutation are “too random” to allow the implementation of a meaningful query strategy without getting some information from the inverse function, even if we allow to “coordinate in advance” the query strategy.

► **Theorem 26**. *Property **Inv**<sup>\*</sup> has a forward-only  $\varepsilon$ -test that uses  $O(\varepsilon^{-2} \log n)$  queries for every  $\varepsilon > 0$ , but any locally-bounded adaptive  $\frac{1}{5}$ -test requires  $\Omega(\sqrt{n})$  queries.*

The upper bound for forward-only testing of **Inv** is demonstrated in Algorithm 2. Applying Algorithm 2 to **Inv**<sup>\*</sup> is pretty straightforward.

■ **Algorithm 2** One sided  $\varepsilon$ -test for **Inv**, forward only,  $O(\varepsilon^{-2})$  queries.

---

```

Treat samples as  $n$ -long strings over  $[n]$ .
let  $s = 1 + \lceil 3\varepsilon^{-2} \rceil$ .
choose  $j_2, \dots, j_s \in [n]$ , uniformly at random and independently.
choose  $k_2, \dots, k_s \in [n]$ , uniformly at random and independently.
query sample 1 at  $j_2, \dots, j_s$ , giving  $f(j_2), \dots, f(j_s)$ .
query sample 1 at  $k_2, \dots, k_s$ , giving  $f(k_2), \dots, f(k_s)$ .
for  $i$  from 2 to  $s$  do
    query sample  $i$  at  $j_i, f(k_i)$ , giving  $g(j_i), g(f(k_i))$ .
    if  $f(j_i) \neq g(j_i)$  and  $g(f(k_i)) \neq k_i$  then
        return REJECT
return ACCEPT

```

---

### The query foresight method

Some adaptive algorithms do not obey the forward only restriction but can be modified to do so, using a method we call *query foresight*. Intuitively, an adaptive algorithm that has some knowledge about the structure of the queries it may make in the future can make them speculatively at present (that is, we make all potential queries to satisfy the forward-only constraint, even though we believe that some of them will later be considered as irrelevant). The more knowledge the algorithm has about the potential future queries, the less queries are wasted on the current sample.

As an example to the query foresight method, we analyze and convert a fully adaptive algorithm for the  $m$ -support property (Algorithm 3)

■ **Algorithm 3** One sided  $\varepsilon$ -test for  $m$ -bounded support, strong  $m + 1$ -memory,  $O(\varepsilon^{-1}m^2)$  queries.

---

**Memory storage for samples:**  $z^1, \dots, z^m; x$ , all initialized to **NULL**.  
**Extra cell:** We have another syntactic “write-only” memory storage  $z^{m+1}$  which we never query.

**take**  $s = 1 + \lceil 2\varepsilon^{-1}m \rceil$  samples.  
**set**  $c, t \leftarrow 0$ .  
**set**  $j_1, \dots, j_m \leftarrow \mathbf{NULL}$   
**for**  $k$  **from** 1 **to**  $s$  **do**

**Invariant 1**  $c = m$  or  $z^{c+1} = \mathbf{NULL}$ .  
**Invariant 2** for  $1 \leq i \leq c$ ,  $z_j^i$  are distinct where  $J = \{j_1, \dots, j_t\}$ .  
**store**  $x \leftarrow$  sample  $k$ .  
**query**  $x$  at  $j_1, \dots, j_t$ , giving substring  $y^k$ .  
**for**  $i$  **from** 1 **to**  $c$  **do**

**query** sample  $z^i$  at  $j_1, \dots, j_t$  giving substring  $y^i$ . ▷ the  $y^i$ s are distinct  
**choose**  $j \in [n]$  uniformly at random.  
**query**  $x$  at  $j$ , giving  $x_j$ .  
**if**  $\exists i : y^i = y^k$  **then** ▷ if exists it is unique  
  **query** sample  $z^i$  at  $j$  giving  $z_j^i$ .  
  **if**  $x_j \neq z_j^i$  **then**  
    **store**  $z^{c+1} \leftarrow x$ .  
    **set**  $j^{t+1} \leftarrow j$ . ▷ keep Invariant 2  
    **set**  $t \leftarrow t + 1$  and  $c \leftarrow c + 1$ . ▷ keep Invariant 1

**else**  
  **store**  $z^{c+1} \leftarrow x$ . ▷ Invariant 2 still holds  
  **set**  $c \leftarrow c + 1$ . ▷ keep Invariant 1

**if**  $c > m$  **then**  
  **return** REJECT

**return** ACCEPT

---

► **Theorem 27.** *Algorithm 3 is a one-sided  $\varepsilon$ -test for being supported by at most  $m$  elements.*

We observe that the general structure of Algorithm 3’s queries is highly predictable, and provide a modified version thereof (Algorithm 4) which is also forward-only, without increasing its worst-case query complexity.

The idea is straightforward: we simulate the run of an adaptive algorithm. Every time that the simulation is about to query a new sample, we make additional speculative queries in the current sample, before dropping it as per the requirement of a forward-only algorithm.

## 45:12 Refining the Adaptivity Notion in the Huge Object Model

If the simulated algorithm makes a query to an old sample, we feed it with the answer of the corresponding speculative query. If such a speculative query does not exist, we either accept (for one-sided algorithms) or behave arbitrarily (for two-sided algorithms). If the prediction is conservative, that is, the speculated queries are ensured to cover all queries to past samples, then the construction guarantees the exact acceptance probability for every individual input. This is not guaranteed when the prediction is not conservative, and in this case we need to analyze the effect of bad speculations.

■ **Algorithm 4** One sided  $\varepsilon$ -test for  $m$ -bounded support, forward only,  $O(\varepsilon^{-1}m^2)$  queries.

---

```

take  $s = 1 + \lceil 2\varepsilon^{-1}m \rceil$  samples.
choose  $j_1, \dots, j_s \in [n]$  uniformly and independently at random.
let  $M$  be an uninitialized  $m \times n$  sparse matrix  $\{0, 1\}$ .    ▷ storage for speculative queries
let  $A$  be an empty list over  $[n]$ .
 $c \leftarrow 0$ .
for  $k$  from 1 to  $s$  do
  Invariant  $M_{i,j}$  is initialized for all  $1 \leq i \leq c$  and  $j \in \{j_1, \dots, j_s\}$ .
  for all  $j$  in  $A$  do                                ▷ simulation of  $y^k$ 
    query sample  $k$  at  $j$ , giving  $x_j^k$ .
  set found  $\leftarrow 0$ .
  for  $i$  from 1 to  $c$  do
    if  $\bigwedge_{j \in A} (M_{i,j} = x_j^k)$  then                    ▷ simulation of the  $y^i$ 's
      set found  $\leftarrow 1$ .
       $j \leftarrow j_k$ .
      query sample  $k$  at  $j$ , giving  $x_j^k$ .
      if  $M_{i,j} \neq x_j^k$  then
         $c \leftarrow c + 1$ .
        add  $j$  to  $A$ .
      query sample  $k$  at  $j_1, \dots, j_s$ , giving  $M_{c,j_1}, \dots, M_{c,j_s}$ . ▷ speculative queries
        ▷ keep the invariant

  if found = 0 then
     $c \leftarrow c + 1$ .
    query sample  $k$  at  $j_1, \dots, j_s$ , giving  $M_{c,j_1}, \dots, M_{c,j_s}$ . ▷ speculative queries
      ▷ keep the invariant

  if  $c > m$  then
    return REJECT
return ACCEPT

```

---

### $k$ -bounded memory algorithms

As per Definitions 14 and 15 we have two models of bounded memory, which we call *weak* and *strong* respectively. Intuitively, in both models, the algorithm gets a stream of samples, and it has an unrestricted access to  $k$  of these samples. When the algorithm needs an access to a new sample, it must give up the ability to access one of the past samples. In the weak model, the algorithm does not have a choice and it must drop the earliest sample. In other words, the weak model has an unrestricted access to a sliding window of the  $k$  most recent samples. In the strong model, the algorithm is allowed to choose the sample to drop.

For  $k = 1$ , the weak and strong models are both equal to each other and to the forward-only model. Intuitively, as  $k$  increases, the algorithm is able to consider more complicated relations between samples, especially  $k$ -ary relations, which are more challenging for  $k - 1$ -memory algorithms.

### Exponential separation from the forward-only model

We use the ability to fully consider binary relations using 2-memory algorithms, compared to the limited ability to do so using forward-only algorithms, to establish an exponential separation between them.

We define a property **Sym** that catches the idea of symmetric functions. For some symmetric function  $f : [m] \times [m] \rightarrow \{0, 1\}$ , a distribution in the property draws a random key  $a \in [m]$  and returns a vector that contains both  $a$  (using a high distance code of length  $m$ ) and all values of  $f$  at points  $(a, b)$  for  $b \in [m]$ . For technical reasons, we use fixed-distance systematic codes to encode  $a$  as a part of the row.

► **Lemma 28** (Systematic code). *There exists a set  $\mathcal{C}$  of error correction codes, such that for every  $n \geq m \geq 10$ , it has a code  $C_{m,n} : [m] \rightarrow \{0, 1\}^n$  with the following properties: (1) Its minimal codeword distance is at least  $\frac{1}{3}$  and (2) The projection of  $C_{m,n}$  on its first  $\lceil \log_2 m \rceil$  bits is one-to-one, that is,  $C_{m,n}$  can be decoded by reading the first  $\lceil \log_2 m \rceil$  bits.*

From now on, every use of systematic codes refers to the set  $\mathcal{C}$  that is guaranteed by Lemma 28, usually denoted just by  $C$  (rather than the explicit notion  $C_{m,n}$ ).

► **Definition 29** (Matrix property **sym**). *For a fixed  $n$ , the property **sym** of functions with two variables  $f : [n]^2 \rightarrow \{0, 1\}$  is defined as the property of being symmetric, i.e. satisfying  $f(i, j) = f(j, i)$  for all  $i, j \in [n]$ .*

The corresponding distribution property is inspired by considering distributions over the rows of a symmetric matrix, along with properly encoded identifiers.

► **Definition 30** (Distribution property **Sym**). *For any  $m$  and the systematic code  $C : [m] \rightarrow \{0, 1\}^m$  from Lemma 28, the property **Sym** is defined as the set of distributions for which*

$$\Pr_{x \sim P} [\exists a \in [m] : x_{1, \dots, m} = C(a)] = 1$$

(all vectors start with an encoding of a “row identifier”), and for every  $a, b \in [m]$ ,

$$\Pr_{x, y \sim P} [x_{1, \dots, m} = C(a) \wedge y_{1, \dots, m} = C(b) \wedge x_{m+b} \neq y_{m+a}] = 0$$

(if two “identifiers”  $a$  and  $b$  appear with positive probability, then the respective “ $f(a, b)$ ” and “ $f(b, a)$ ” are identical).

► **Theorem 31**. *There exists a one-sided weak 2-memory  $\varepsilon$ -test for **Sym** that makes  $O(\varepsilon^{-2} \log n)$  queries, but every forward-only  $\frac{1}{14}$ -test for **Sym** must use at least  $\Omega(\sqrt{m})$  queries (for sufficiently large  $m$ ).*

The lower bound follows from a forward-only algorithm being given access to every sample without any knowledge about the keys of “future” samples. If the algorithm has only one accessible sample at a time, it can only “guess” the other key, but the probability to actually draw a later sample with that key is too low, unless the algorithm collects queries according to about  $\sqrt{m}$  guessed keys.

## 45:14 Refining the Adaptivity Notion in the Huge Object Model

For the upper-bound, Algorithm 5 performs a sequence of independent iterations using two samples at a time. In every iteration, it gathers their “keys”  $a_1$  and  $a_2$ , verifies the correctness of their codewords, and then checks whether  $f(a_1, a_2) = f(a_2, a_1)$ . There are some cases that should be carefully analyzed, for example the case where the distribution does not correspond to a single  $f$ , or the case where some values for “ $a$ ” appear very rarely or not at all, but these do not defeat the above algorithm (they somewhat affect its number of needed iterations).

■ **Algorithm 5** One-sided  $\varepsilon$ -test for **Sym**, weak 2-memory,  $O(\varepsilon^{-2} \log n)$  queries.

---

```

let  $m \leftarrow n/2$ .
for  $\lceil 8\varepsilon^{-2} \rceil$  times do
  take two samples  $x, y$ .
  query  $x_1, \dots, x_{\lceil \log_2 m \rceil}$ , giving  $\kappa(x)$  as  $a$ .
  query  $y_1, \dots, y_{\lceil \log_2 m \rceil}$ , giving  $\kappa(y)$  as  $b$ .
  choose  $i \in [m]$ , uniformly at random.
  query  $x, y$  at  $i$ , giving  $x_i, y_i$ .
  query  $\phi_x(b), \phi_y(a)$ .
  if  $x_i \neq (C(a))_i$  or  $y_i \neq (C(b))_i$  then
    return REJECT ▷ rejection by key invalidity
  if  $\phi_x(b) \neq \phi_y(a)$  then
    return REJECT ▷ rejection by asymmetry
return ACCEPT

```

---

### Larger memory generalization

We generalize the above theorem to state an exponential separation between the  $k$ -weak model (Definition 14) and the  $k-1$ -strong model (Definition 15). We define a property **Par<sub>k</sub>** based on parity for every  $k \geq 2$  for which:

► **Theorem 32.** *For every  $k \geq 2$ , there exists a one-sided weak  $k$ -memory  $\varepsilon$ -test for **Par<sub>k</sub>** that makes  $O(k\varepsilon^{-k} \log n)$  queries, but every forward-only  $\frac{1}{6k}$ -test for **Par<sub>k</sub>** must use at least  $\Omega(\sqrt{m})$  (for sufficiently large  $n$ ), which is  $\Omega(n^{1/2k})$  queries since  $n \approx \binom{m}{k-1}$ .*

To motivate the definition of **Par<sub>k</sub>**, suppose that  $f : \binom{[m]}{k} \rightarrow \{0, 1\}^k$  is a function such that  $f(A)$  has zero parity for every subset  $A \subseteq [m]$  of size  $k$ . We “encode” such a function as a distribution, making sure to “separate” the  $k$  bits of  $f(A)$  to  $k$  different samples. A typical sample in the distribution would have an encoding (using a high distance code) of a random key  $a \in [m]$ , followed by some information on  $f(A)$  for every  $A$  that contains  $a$ . Specifically, for each such  $A$  we supply the  $i$ th bit of  $f(A)$ , where  $i$  is the “rank” of  $a$  in  $A$  (going by the natural order over  $[m]$ ).

► **Definition 33** (Preliminaries for distribution property **Par<sub>k</sub>**). *Let  $k \geq 2$  be the degree of freedom in the represented function. Let  $m$  be the (sufficiently large) size of the input set and  $n = \binom{m-1}{k-1}$ . Also, consider a systematic code  $C : [m] \rightarrow \{0, 1\}^n$ .*

*For a string  $x \in \{0, 1\}^{2n}$ , let  $\kappa(x) = C^{-1}(x_{1, \dots, n})$  be the key of  $x$  (if the inverse function is not defined we can use an arbitrary key instead). Since we have an implicit mapping between  $k-1$ -subsets of  $\{1, \dots, m\} \setminus \{\kappa(a)\}$  and the indexes  $\{1, \dots, n\}$ , for every  $A \subseteq \{1, \dots, m\} \setminus \{\kappa(a)\}$  of size  $k-1$  we can define  $\Phi_A(x)$  as the corresponding bit in  $x_{n+1, \dots, 2n}$ .*

► **Definition 34** (Distribution property **Par<sub>k</sub>**). *For  $k, m, n, C$  defined above, the property **Par<sub>k</sub>** is defined as the set of distributions over  $\{0, 1\}^{2n}$  for which*

$$\Pr_{x \sim P} [\exists a \in [m] : x_{1, \dots, n} = C(a)] = 1$$



(all vectors start with an encoding of a “row identifier”), and for every  $a_1 < \dots < a_k \in [m]$ ,

$$\Pr_{x^1, \dots, x^k \sim P} \left[ \bigwedge_{i=1}^k (x^i)_{1, \dots, n} = C(a_i) \wedge \bigoplus_{i=1}^k \Phi_{\{a_1, \dots, a_k\} \setminus \{a_i\}}(x^i) = 1 \right] = 0$$

(if all “identifiers”  $a_1, \dots, a_k$  appear with positive probability, then the respective concatenation of values which forms “ $f(\{a_1, \dots, a_k\})$ ” has zero parity).

For the lower bound, if the algorithm has less than  $k$  accessible samples at a time, as with the analysis of the **Sym** property under forward-only testing, the algorithm here can only “guess” the missing key, and the probability to make the right guess is too low.

We go further, and show that even if the  $k - 1$ -memory algorithm is allowed to choose which of the samples are retained in every stage (strong  $k - 1$ -memory) rather than keeping a sliding window of recent history, the exponential separation still holds. The separation is achieved for an  $\varepsilon_k$ -test of the property where  $\varepsilon_k = \Theta(1/k)$ .

For the upper bound, Algorithm 6 makes a sequence of independent iterations of  $k$  samples at a time. In every iteration it gathers the keys  $a_1, \dots, a_k$  and verifies their codewords. If they are all different, the algorithm constructs the value of  $f(\{a_1, \dots, a_k\})$  and checks its parity.

■ **Algorithm 6** One-sided  $\varepsilon$ -test for **Par** $_k$ , weak  $k$ -memory,  $O(\varepsilon^{-k} k \log n)$  queries.

---

```

let  $m$  be such that  $\binom{m-1}{k-1} = n$ .
for  $\lceil 4\varepsilon^{-k} k \rceil$  times do
  take  $k$  new samples  $x^1, \dots, x^k$ .
  for  $t$  from 1 to  $k$  do
    query  $x^1, \dots, x^{\lceil \log_2 m \rceil}$ , giving  $\kappa(x^t)$  as  $a^t$ .
    choose  $i \in [m]$ , uniformly at random.
    query  $x^t$  at  $i$ , giving  $x_i^t$ .
    if  $x_i^t \neq (C(a^t))_i$  then
      return REJECT ▷ reject by key invalidity
  if  $|\{a^1, \dots, a^k\}| = k$  then
    for  $t$  from 1 to  $k$  do
      query  $\Phi_{x^t}(\{a^1, \dots, a^k\} \setminus \{a^t\})$ , giving  $s^t$ .
    if  $\bigoplus_{i=1}^k s^t = 1$  then
      return REJECT ▷ reject by parity-invalidity
return ACCEPT

```

---

## 4 Remaining open problems

It is an open problem whether the weak  $k$ -memory model is indeed strictly weaker than the strong  $k$ -memory model (for the same  $k$ ). And if so, is the separation exponential? Also, we do not know whether or not for every  $k$  there exists  $k^*$  such that the  $k^*$ -weak model contains the  $k$ -strong one.

We believe that there exist some  $\varepsilon_0 > 0$  and  $0 < \alpha < 1$  such that for every sufficiently large  $k$ , there is an exponential separation between the weak  $k$ -memory model and the strong  $\alpha k$ -memory model, with respect to an  $\varepsilon_0$ -test, rather than the separation for  $\varepsilon_k = \Theta(1/k)$  that we show for  $k - 1$  vs  $k$  memory.

Another interesting open problem is whether the fully adaptive model has a simultaneous exponential separation from all fixed  $k$ -memory models. That is, whether there exists a property  $\mathcal{P}$  and some  $\varepsilon_0 > 0$  such that  $\varepsilon_0$ -testing of  $\mathcal{P}$  would require  $\Omega(\text{poly}(n))$  queries in every  $k$ -memory model (the polynomial degree possibly depending on  $k$ ), but  $\mathcal{P}$  is  $\varepsilon$ -testable using  $O(\log n)$  queries using a fully adaptive algorithm for every fixed  $\varepsilon > 0$ .

---

## References

- 1 Tomer Adar, Eldar Fischer, and Amit Levi. Support testing in the huge object model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London, United Kingdom*, volume 317. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 2 Maryam Aliakbarpour, Andrew McGregor, Jelani Nelson, and Erik Waingarten. Estimation of entropy in constant space with improved sample complexity. In *Proceedings of the 34th Annual Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- 3 Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30(6):1842–1862, 2001.
- 4 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001.
- 5 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE, 2000.
- 6 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 73–83, 1990.
- 7 Clément L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Number 9 in Graduate Surveys. Theory of Computing Library, 2020. doi:10.4086/toc.gs.2020.009.
- 8 Sourav Chakraborty, Eldar Fischer, Arijit Ghosh, Gopinath Mishra, and Sayantan Sen. Testing of index-invariant properties in the huge object model. *CoRR*, abs/2207.12514, 2022. doi:10.48550/arXiv.2207.12514.
- 9 Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate pcps. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 41–50, 1999.
- 10 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.
- 11 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75, 2011.
- 12 Oded Goldreich and Dana Ron. Testing distributions of huge objects. 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 78:1–78:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ITCS.2022.78.
- 13 David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.
- 14 Ronitt Rubinfeld and Madhu Sudan. Self-testing polynomial functions efficiently and over rational domains. In *SODA*, pages 23–32, 1992.
- 15 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.
- 16 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

# Support Testing in the Huge Object Model

Tomer Adar  

Technion – Israel Institute of Technology, Haifa, Israel

Eldar Fischer 

Technion – Israel Institute of Technology, Haifa, Israel

Amit Levi  

University of Haifa, Israel

---

## Abstract

The Huge Object model is a distribution testing model in which we are given access to independent samples from an unknown distribution over the set of strings  $\{0, 1\}^n$ , but are only allowed to query a few bits from the samples. We investigate the problem of testing whether a distribution is supported on  $m$  elements in this model. It turns out that the behavior of this property is surprisingly intricate, especially when also considering the question of adaptivity.

We prove lower and upper bounds for both adaptive and non-adaptive algorithms in the one-sided and two-sided error regime. Our bounds are tight when  $m$  is fixed to a constant (and the distance parameter  $\epsilon$  is the only variable). For the general case, our bounds are at most  $O(\log m)$  apart. In particular, our results show a surprising  $O(\log \epsilon^{-1})$  gap between the number of queries required for non-adaptive testing as compared to adaptive testing. For one-sided error testing, we also show that an  $O(\log m)$  gap between the number of samples and the number of queries is necessary. Our results utilize a wide variety of combinatorial and probabilistic methods.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Huge-Object model, Property Testing

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.46

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2308.15988>

**Funding** *Eldar Fischer*: Research supported by an Israel Science Foundation grant number 879/22.

## 1 Introduction

Property testing [12, 7] is a framework concerned with analyzing global properties of an input while reading only a small part thereof, in the form of queries. Over the past few decades property testing has become an active field of study in theoretical computer science (see e.g., [6]). The study of distribution property testing was first implicitly explored in [8], and explicitly formulated in [3] and [4]. In the standard model of distribution testing, an algorithm can access a sequence of independently sampled elements drawn from an unknown input distribution  $\mu$ , and it either accepts or rejects the input based on this sequence. An  $\epsilon$ -testing algorithm for a property of distributions is required to accept every input distribution that satisfies the property with high probability (e.g.,  $\frac{2}{3}$ ), and to reject with high probability (e.g.,  $\frac{2}{3}$ ) every input distribution whose variation distance from every distribution satisfying the property is greater than  $\epsilon$ .

The standard model of distribution testing assumes that the elements drawn from the distribution are fully accessible, which might be unreasonable if they have very large descriptions (“huge objects”). The Huge Object model, whose study was initiated in [9], treats the sampled elements as long strings that have to be queried. In this model, for



© Tomer Adar, Eldar Fischer, and Amit Levi;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 46; pp. 46:1–46:16



Leibniz International Proceedings in Informatics

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

example, it is possible that the algorithm has two non-identical samples without being able to distinguish between them efficiently. This “two-phase” characteristic of the Huge Object model (“sample then query”, rather than only taking samples or only querying a string) exhibits rich behavior with respect to adaptive querying, as studied in detail in [1].

In the standard model of distribution testing, [13] and [14] show a tight bound of  $\Theta(m/\log m)$  samples for two-sided error  $\varepsilon$ -testing of having a support size bounded by  $m$  in the standard model, for every fixed  $\varepsilon$ . An upper bound of  $O(\varepsilon^{-1}m)$  samples for one-sided algorithms is implicitly shown in [1], and here we show that it is tight (Proposition 24). Based on these tight bounds, the bounded support property is considered to be fully understood in the standard model for one-sided testing, and mostly understood in the two-sided case (for every fixed  $m$  there is still a gap between  $\Omega(\varepsilon^{-1})$  and  $O(\varepsilon^{-2})$  for two-sided testing).

One would expect that having bounded support, which is arguably the simplest of distribution properties, would have simple and easily understood testing bounds also in the Huge Object model. As in the standard model, it is the only label-invariant property that is testable using one-sided error algorithms. However, it turns out that the behaviour of this property under the Huge Object model is surprisingly intricate. One unexpected feature that we show here is a gap between the number of queries required for non-adaptively testing for this property as compared to adaptive testing. Indeed there is no adaptivity in the standard distribution testing model, and one would not expect the label-invariant (and even mapping-invariant as per the definition in [9]) property of having bounded support to exhibit such a gap.

## 1.1 Definition of the model

The Huge Object model differs from the standard sampling model in its distance measure and in the way that the algorithm gathers information about the input distribution.

### Algorithmic model

A probabilistic algorithm  $\mathcal{A}$  with  $q$  queries and  $s$  samples, whose input is a distribution  $P$  over  $\{0, 1\}^n$  accessible via the Huge Object model, is an algorithm that acts in the following manner: at every stage, the algorithm may ask for a new sample  $v$  that is provided by drawing it according to  $P$ , independently of all prior samples, or may ask to query a coordinate  $j \in \{1, \dots, n\}$  of an old sample  $u$  (the algorithm may use internal coin tosses to make its decisions). When this query is made, the algorithm is provided with  $u_j \in \{0, 1\}$  as its answer. The algorithm has no access to the sampled vectors apart from query access. In the end, after taking not more than a total of  $s$  samples and making a total of not more than  $q$  queries, the algorithm provides its output.

We say that the algorithm is *non-adaptive* if it makes all its sampling and querying decisions in advance, prior to receiving all query answers in bulk. Only the final output of a non-adaptive algorithm may depend on the received answers.

### Distances

Here we define some measures of distance. Note that we usually use  $d(\cdot, \cdot)$  without mentioning the measure that we use, if its context is unambiguous. For distributions over  $\{0, 1\}^n$ ,  $d(\cdot, \cdot)$  usually refers to the earth mover’s distance defined below.

► **Definition 1** (String distance). Let  $u, v \in \{0, 1\}^n$  be two strings. We define their distance as the normalized Hamming distance,

$$d_H(u, v) = \frac{1}{n} |\{1 \leq i \leq n \mid u_i \neq v_i\}| = \Pr_{i \sim \{1, \dots, n\}} [u_i \neq v_i]$$

We define the distance of  $u \in \{0, 1\}^n$  from a set  $A \subseteq \{0, 1\}^n$  as  $d_H(u, A) = \min_{v \in A} d_H(u, v)$ .

► **Definition 2** (Transfer distribution). Let  $P$  and  $Q$  be distributions over finite sets  $\Omega_1$  and  $\Omega_2$ , respectively. A distribution  $T$  over  $\Omega_1 \times \Omega_2$  is a transfer distribution from  $P$  to  $Q$  if for every  $a \in \Omega_1$ ,  $\Pr_{(u,v) \sim T} [u = a] = P(a)$ , and for every  $b \in \Omega_2$ ,  $\Pr_{(u,v) \sim T} [v = b] = Q(b)$ . The set of transfer distributions from  $P$  to  $Q$  is denoted by  $\mathcal{T}(P, Q)$ . Note that this is a compact set when considered as a set of real-valued vectors.

► **Definition 3** (Variation distance). Let  $\mu$  and  $\nu$  be two distributions over a finite set  $\Omega$ . Their variation distance is defined as:

$$d_{\text{var}}(\mu, \nu) = \frac{1}{2} \sum_{u \in \Omega} |\mu(u) - \nu(u)| = \max_{E \subseteq \Omega} \left| \Pr_{\mu} [E] - \Pr_{\nu} [E] \right| = \min_{T \in \mathcal{T}(\mu, \nu)} \Pr_{(u,v) \sim T} [u \neq v]$$

► **Definition 4** (Earth mover's distance). Let  $P$  and  $Q$  be two distributions over  $\{0, 1\}^n$ . Their earth mover's distance is defined as:

$$d_{\text{EMD}}(P, Q) = \min_{T \in \mathcal{T}(P, Q)} \mathbb{E}_{(u,v) \sim T} [d_H(u, v)]$$

The above minimum exists since it is in particular the minimum of a continuous function over a compact set.

## Testing model

► **Definition 5** (A property). A property  $\mathcal{P}$  is a sequence  $\mathcal{P}_1, \mathcal{P}_2, \dots$  such that for every  $n \geq 1$ ,  $\mathcal{P}_n$  is a compact subset of the set of all distributions over  $\{0, 1\}^n$ .

► **Definition 6** (Distance of a distribution from a property). Let  $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \dots)$  be a property and  $P$  be a distribution over  $\{0, 1\}^n$  for some  $n$ . The distance of  $P$  from  $\mathcal{P}$  is defined as  $d_{\text{EMD}}(P, \mathcal{P}) = \min_{Q \in \mathcal{P}_n} \{d_{\text{EMD}}(P, Q)\}$ .

► **Definition 7** ( $\varepsilon$ -test). Let  $\mathcal{P}$  be a property of distributions over  $\{0, 1\}^n$ . We say that a probabilistic algorithm  $\mathcal{A}$  is an  $\varepsilon$ -test for  $\mathcal{P}$  if:

- For every  $P \in \mathcal{P}$ ,  $\mathcal{A}$  accepts with probability higher than  $\frac{2}{3}$ .
- For every probability distribution  $P$  over  $\{0, 1\}^n$  that is  $\varepsilon$ -far from  $\mathcal{P}$  (satisfying  $d(P, \mathcal{P}) > \varepsilon$ ),  $\mathcal{A}$  rejects with probability higher than  $\frac{2}{3}$ .

► **Definition 8** (one-sided and two-sided  $\varepsilon$ -test). Consider the setting of the above definition. If additionally for every input  $P \in \mathcal{P}$ ,  $\mathcal{A}$  accepts  $\mathcal{P}$  with probability 1 (rather than “higher than  $\frac{2}{3}$ ”), then we say that  $\mathcal{A}$  is a one-sided  $\varepsilon$ -test for  $\mathcal{P}$ . Otherwise, we say that  $\mathcal{A}$  has two-sided error.

## 1.2 Summary of our results

### Table of results

The following is a table summarizing the bounds presented here for  $\varepsilon$ -testing for being supported by at most  $m$  elements, along with previously known ones provided for reference (Section 3 contains a sketch on how to derive them). The hidden coefficients in the  $O(\cdot)$  and the  $\Omega(\cdot)$  notations are global numerical constants. The new results appear in purple.

## 46:4 Support Testing in the Huge Object Model

| Model                                 | One-sided Error  | Two-sided Error   |
|---------------------------------------|--|---|
| Standard model<br>(Sample complexity) | $\Theta(\varepsilon^{-1}m)$<br>Folklore, see [1]   | $\Omega(\varepsilon^{-1}m/\log m)$ [13]<br>$O(\varepsilon^{-2}m/\log m)$ [14]                               |
| Huge Object<br>Non-adaptive           | $\Omega(\varepsilon^{-1}m(\log \varepsilon^{-1} + \log m))$<br>$O(\varepsilon^{-1}m \log \varepsilon^{-1} \log m)$ | $\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$<br>$O(\varepsilon^{-3}m \log \varepsilon^{-1})$ [14] + [9] |
| Huge Object<br>Adaptive               | $\Omega(\varepsilon^{-1}m \log m)$<br>$O(\varepsilon^{-1}m \log m \cdot \min\{\log \varepsilon^{-1}, \log m\})$    | $\Omega(\varepsilon^{-1}m/\log m)$ [13]   |

The following are some conclusions to be drawn from the bounds given above. We use  $\mathcal{S}_m$  to denote the property of being supported by at most  $m$  elements.

### Adaptive vs. non-adaptive two-sided asymptotic gap

The most surprising result is that non-adaptively testing a distribution for being supported by at most two elements cannot be done using a number of queries linear in  $\varepsilon^{-1}$ , even with two-sided error. This result applies for every  $m \geq 2$ , and the exact bound is  $\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$  (with the implicit coefficient being independent of  $m$ ). To the best of our knowledge, combined with the  $O(\varepsilon^{-1})$  adaptive upper bound of [1] (which we improve in this paper), “being supported by at most two elements” is the first explicit example of a property that is closed under mapping (and in particular is label-invariant) which has different asymptotic bounds for the number of queries for adaptive algorithms and non-adaptive ones in the Huge Object model (see Theorem 26).

A possible explanation for this is that being label-invariant in the Huge Object model is different from being so in the standard model, because applying a permutation on the labels may change their distinguishability, and in particular it may change the distance from the property.

In this paper we provide a thorough investigation of  $\mathcal{S}_m$  utilizing a variety of methods. In particular, we show several gaps such as the above mentioned one. However, the behaviour of the bounded support property in the Huge Object model, especially when considering it as a problem with two variables (namely the maximal support sized  $m$  and the distance parameter  $\varepsilon$ ) is still not completely understood. We do have tight bounds for the fixed constant  $m$  cases (where only  $\varepsilon$  is variable) for all algorithm types, and bounds up to logarithmic factors for the more general cases.

### One-sided bounds and a gap from the standard model

We have tight bounds for  $\varepsilon$ -testing of  $\mathcal{S}_m$  for every fixed  $m$  (and variable  $\varepsilon$ ) for both non-adaptive algorithms and adaptive ones. These bounds are also tight for every fixed  $\varepsilon$  (and variable  $m$ ). Additionally, our bounds show a gap between the standard model (considering sample complexity) and the Huge Object model (considering query complexity). Consider the bounded support property as a sequence of individual properties, where for every  $m \geq 2$ , the  $m$ -th property is  $\mathcal{S}_m$ . We show that, if we only allow one-sided error tests, there is an  $O(\log m)$  gap between the standard model of distribution testing and the Huge Object model. In the standard model, there exists a one-sided test for  $\mathcal{S}_m$  at the cost of  $O(\varepsilon^{-1}m)$  samples. In the Huge Object model, there is a lower bound of  $\Omega(\varepsilon^{-1}m \cdot \log m)$  many queries for every one-sided  $\varepsilon$ -test, even if it is adaptive. Note that the gap is between the number of *samples* in the standard model and the number of *queries* in the Huge Object model, which is the natural measure of complexity in this model.

## New tools

### A new algorithmic paradigm

For the adaptive one-sided upper bound, we define a standalone algorithmic primitive, the “fishing expedition” paradigm, that repeatedly executes a subroutine until it reaches a predefined goal or when it finds out that it is no longer cost-effective (even if it did not reach the goal). We believe that this primitive will also be useful in future endeavors.

### A hybrid probabilistic-extremal analysis

We define a concept of “valid composition”. Loosely speaking, it is an ordered subset of samples that become closer to each other as the sequence progresses, but are still  $\varepsilon$ -far from each other. We use a hybrid probabilistic-extremal argument to show that for an input distribution that is  $\varepsilon$ -far from  $m$ -support, with high probability, an algorithm with a bounded number of queries will find a valid composition with at least  $m + 1$ -elements.

The hybrid probabilistic-extremal argument works as follows: we define some rank of valid compositions. If for every individual valid composition with at most  $m$  elements, there is a high probability that it is not maximal (according to the rank), then globally there is a high probability that none of them is maximal. Hence, with high probability, the maximally-ranked valid composition within our samples must have at least  $m + 1$  elements.

### A new use for an old combinatorial result

For the adaptive one-sided lower bound, we use an old combinatorial result, that a biclique covering of the  $m$ -clique must have at least  $m \log_2 m$  vertices [10, 11], to show that every witness against  $m$ -support is at least  $m \log m$  bits long, which makes it a lower bound to the number of queries. To apply a multiplicative factor of  $\varepsilon^{-1}$ , which is pretty easy for non-adaptive algorithms, in adaptive algorithms we analyze the effectivity of a decision tree that incrementally constructs a witness based on the queries.

## 1.3 Open problems

### One-sided non-adaptive bounds

We have an  $\Omega(\varepsilon^{-1}m(\log \varepsilon^{-1} + \log m))$  lower bound for one-sided  $\varepsilon$ -testing of  $\mathcal{S}_m$ , as well as an  $O(\varepsilon^{-1}m \log \varepsilon^{-1} \log m)$  upper bound for one-sided  $\varepsilon$ -testing of  $\mathcal{S}_m$ . We believe that the upper bound is tight, but we do not have the corresponding lower bound. What is the true complexity of one-sided  $\varepsilon$ -testing  $\mathcal{S}_m$ ?

### Non-trivial two-sided bounds

Is there a lower bound of  $\omega(m/\log m)$  queries for two-sided testing of  $\mathcal{S}_m$  (noting that [13] only gives  $\Omega(m/\log m)$ ), even for non-adaptive algorithms? We believe that  $\Omega(m)$  should be this lower bound, based on the  $\log m$  gap in the one-sided case (a  $\Theta(m)$  tight bound in the standard model, and a  $\Theta(m \log m)$  tight bound in the Huge Object model).

### One-sided adaptive bounds

Our results for one-sided adaptive  $\varepsilon$ -testing of  $\mathcal{S}_m$  are tight with respect to  $m$ , but have a logarithmic gap with respect to  $\min\{\varepsilon^{-1}, m\}$ . Closing this gap is an open problem.

### The tradeoffs between sample and query complexity

Our bounds apply to the query complexity of the tests. The lower bounds adapted from previous works on the traditional model clearly apply for the sample complexity here, even if we allow a higher query complexity. As for our new upper bounds, most of them have a polylogarithmic average queries per sample ratio. It would be interesting to investigate whether the sample complexity can be reduced if we allow a much higher (but still sub-linear in  $n$ ) number of queries per sample.

## 2 Preliminaries

### 2.1 Algorithmic model

As observed by Yao [16], every probabilistic algorithm can be seen as a distribution over a set of deterministic algorithms. Hence we can analyze probabilistic query-making algorithms by analyzing the deterministic algorithms they are supported on.

We observe that we can assume that all samples are drawn before the first query is made, since they are fully independent: the distribution of every sample made does not depend at all on any calculation or queries that occurred before it was taken, and so we can assume that it was taken before any calculation was performed. Based on this observation we can represent our algorithms using a  $\{0, 1\}$ -valued matrix (whose rows are sampled from the distribution), from which the algorithms are allowed to query.

► **Definition 9** (Matrix representation of input access). *Considering an algorithm with  $s$  samples and  $q$  queries, we assume that the samples are all taken at the beginning of the algorithm and are used to populate a matrix  $M \in \{0, 1\}^{s \times n}$ . Then, during the run of the algorithm, each of its queries is represented as a pair  $(i, j) \in \{1, \dots, s\} \times \{1, \dots, n\}$ , for which the answer is  $M_{i,j}$ .*

► **Definition 10** (Adaptive algorithm). *Every deterministic algorithm in the Huge Object model with  $q$  queries over  $s$  samples is equivalent to a pair  $(T, A)$ , where  $T$  is a decision tree of height  $q$  in which every internal node contains a query  $(i, j)$  (where  $1 \leq i \leq s$  is the index of a sample and  $1 \leq j \leq n$  is the index to query), and  $A$  is the set of accepting leaves.*

► **Definition 11** (Non-adaptive algorithm). *A deterministic algorithm  $(T, A)$  with  $q$  queries is non-adaptive if, for every  $0 \leq i < q$ , all internal nodes at the  $i$ -th level consist of the exact same query. Every non-adaptive algorithm can be represented as a pair  $(Q, A)$ , where  $Q \subseteq \{1, \dots, s\} \times \{1, \dots, n\}$  is a set of queries and  $A \subseteq \{Q \mapsto \{0, 1\}\}$  is the set of accepted answer vectors.*

### 2.2 Technical components

#### Fishing expedition

We define an algorithmic primitive that allows us to repeat an execution of a probabilistic subroutine until it is no longer effective. Consider for example a “coupon-collector” type process, but one in which the number of distinct elements is not known to us. The goal is to collect a preset number of elements, but we also want to stop early if we believe that there are no more elements to be effectively collected.

Consider a (probabilistic) subroutine  $\mathcal{A}$  that can either fail or succeed. We denote the outcome of an execution of  $\mathcal{A}$  by  $R$ . In this discussion the outcome includes both the explicit output of the execution and its side effects, which may affect the probabilities for



future executions of  $\mathcal{A}$ . We thus analyze a *sequence* of executions  $R_1, \dots, R_N$ , where  $R_1$  is performed over the initial state. We define two behaviors of “coupon collection” that such an  $\mathcal{A}$  must present.

► **Definition 12** (Fail stability). *Let  $\mathcal{A}$  be a subroutine that may succeed or fail. Specifically let  $R_1, \dots, R_N$  be random variables that detail the outputs of the first  $N$  executions of  $\mathcal{A}$ . We say that  $\mathcal{A}$  is fail stable with respect to a set  $G$  of outcomes indicating success, if for every  $2 \leq i \leq N$  and every result sequence  $(r_1, \dots, r_{i-1}) \in \text{supp}(R_1, \dots, R_{i-1})$  for which  $r_{i-1} \notin G$ :*

$$\begin{aligned} \Pr[R_i \in G \mid R_1 = r_1, \dots, R_{i-2} = r_{i-2}, R_{i-1} = r_{i-1}] \\ = \Pr[R_{i-1} \in G \mid R_1 = r_1, \dots, R_{i-2} = r_{i-2}] \end{aligned}$$

*In other words, a failure does not affect the probability of further executions to succeed.*

► **Definition 13** (Diminishing returns). *Let  $\mathcal{A}$  and  $R_1, \dots, R_N$  be as in Definition 12. We say that  $\mathcal{A}$  has diminishing returns with respect to a set  $G$  of successful outcomes, if for every  $2 \leq i \leq N$  and every result sequence  $(r_1, \dots, r_{i-1}) \in \text{supp}(R_1, \dots, R_{i-1})$ :*

$$\begin{aligned} \Pr[R_i \in G \mid R_1 = r_1, \dots, R_{i-2} = r_{i-2}, R_{i-1} = r_{i-1}] \\ \leq \Pr[R_{i-1} \in G \mid R_1 = r_1, \dots, R_{i-2} = r_{i-2}] \end{aligned}$$

*That is, if  $\mathcal{A}$  has diminishing returns, then a success in a single execution never increases, but may decrease, the probability of further executions to succeed.*

Recall the coupon-collecting example. We expect it to have both fail stability and diminishing returns (with respect to a common set  $G$  of outcomes indicating success). If we look for a coupon and do not find it in a single try, nothing happens. Further tries will have the same probability to succeed. On the other hand, if we collect a coupon, then in further tries, there are less uncollected coupons left and it is slightly harder to find an additional one.

The fishing expedition paradigm seeks to collect a goal of  $k$  coupons, but “gives up” if it believes that the probability to find an additional coupon is less than some parameter  $p$ .

The desired algorithm (Algorithm 1) has three parameters: a threshold  $p$ , a confidence  $q$  and a goal  $k \geq 1$ . The input is a subroutine  $\mathcal{A}$  with diminishing returns and fail stability (with respect to some common set  $G$ ). Informally, the goal of the algorithm is to have  $k$  successful executions of  $\mathcal{A}$ , but also to terminate earlier if the probability of  $\mathcal{A}$  to succeed becomes lower than  $p$ . Since the algorithm has no actual access to the success probability of  $\mathcal{A}$ , it should terminate early only if it is confident enough that the success probability of further executions is too low for them to be effective.

► **Lemma 14.** *Consider a black box subroutine  $\mathcal{A}$  with fail stability (Definition 12) and diminishing returns (Definition 13) with respect to a common set  $G$  of outcomes indicating success.*

*For an algorithm that repeatedly executes  $\mathcal{A}$ , we define the following random variables:*

- $N$  – the number of executions.
- $R_1, \dots, R_N$  – their outcomes.
- $X_1, \dots, X_N$  – indicators of success (that is,  $X_i = 1$  if and only if  $R_i \in G$ ).
- $H = \sum_{i=1}^N X_i$  – the number of successful executions.
- $\hat{p} = \Pr[X_{N+1} = 1 \mid R_1, \dots, R_N]$  – the success probability of a possible extra execution of  $\mathcal{A}$ .

*Considering the parameters  $p > 0$  (threshold),  $q > 0$  (confidence), and  $k \geq 1$  (goal), there exists an algorithm that repeatedly executes  $\mathcal{A}$  for which  $N \leq p^{-1}(4H + 5(\log q^{-1} + \log(\log k + 1))) + 1$  and  $H \leq k$ , such that with probability higher than  $1 - q$ , either  $H = k$  or  $\hat{p} \leq p$  (or both).*

■ **Algorithm 1** Fishing expedition.

---

**parameters**  $k \geq 1$  (goal),  $p > 0$  (threshold),  $q > 0$  (confidence).  
**input** A subroutine  $\mathcal{A}$  with output, given as a black box, where an output outside a set  $G$  means FAIL.  
**let**  $t_{\max} \leftarrow \lfloor \log k + 1 \rfloor$ .  
**let**  $N_1 \leftarrow 0$ .  
**set**  $H \leftarrow 0$ .  
**for**  $t$  **from** 2 **to**  $t_{\max}$  **do**  
  **let**  $N_t \leftarrow \lceil p^{-1} \max\{2^t, 5(\log q^{-1} + \log(\log k + 1))\} \rceil$ .  
  **for**  $N$  **from**  $N_{t-1} + 1$  **to**  $N_t$  **do** ▷ possibly empty  
    **run**  $\mathcal{A}$ , let  $R_N$  be its outcome.  
    **let**  $X_N$  be an indicator for success ( $X_N = 1$  if  $R_N \in G$ , otherwise  $X_N = 0$ ).  
    **set**  $H \leftarrow H + X_N$ .  
    **if**  $H = k$  **then terminate** with  $N$ . ▷ goal is reached  
  **if**  $H < \frac{1}{2}pN_t$  **then**  
    **terminate** with  $N_t$ . ▷ continuing is ineffective

---

The proof of the lemma follows from two claims. The first claim asserts that for  $t_{\max} = \lfloor \log k + 1 \rfloor$  and for every  $2 \leq t \leq t_{\max}$ , after  $\lceil p^{-1} \cdot \max\{2^t, 5(\log q^{-1} + \log(\log k + 2))\} \rceil$  executions of  $\mathcal{A}$ , the algorithm terminates if the number of successful executions was less than a  $\frac{1}{2}p$ -portion of the total number of executions. The second claim shows that the algorithm reaches one of its goals with probability higher than  $1 - q$ , and uses a variant of Chernoff's inequality to give an upper bound on the probabilities of bad events.

### Contradiction graph

We define here what it means to be a “counter-example” for having support size at most  $m$ .

► **Definition 15** (Contradiction graph). *Let  $x_1, \dots, x_s \in \{0, 1\}^n$  be a sequence of strings. Let  $Q \subseteq \{1, \dots, s\} \times \{1, \dots, n\}$  be a set of queries. We define the contradiction graph of  $(x_1, \dots, x_s; Q)$  as  $G(V, E)$  with  $V = \{1, \dots, s\}$ , and for every  $1 \leq i_1, i_2 \leq s$ :*

$$\{i_1, i_2\} \in E \iff \exists 1 \leq j \leq n : (x_{i_1})_j \neq (x_{i_2})_j \wedge ((i_1, j), (i_2, j) \in Q)$$

*Note that the graph is undirected since the definition of the edges is commutative. It is also clearly without self-loops.*

► **Definition 16** (Witness against  $m$ -support). *Let  $P$  be a distribution that is supported by a set of more than  $m$  elements. We say that  $(x_1, \dots, x_s; Q)$  is a witness against  $m$ -support (of  $P$ ) if  $x_1, \dots, x_s$  are all drawn from  $P$ , and their contradiction graph is not  $m$ -colorable.*

In the full version, we prove that calling the above a witness is indeed justified, in the sense that a distribution  $P$  has  $m$ -support if and only if there is zero probability to draw a tuple  $x_1, \dots, x_s$  for which one can provide a query set  $Q$  that makes it a witness.

► **Lemma 17.** *Let  $x_1, \dots, x_s \in \text{supp}(P)$  be a set of samples and let  $Q \subseteq \{1, \dots, s\} \times \{1, \dots, n\}$  be a query set. Let  $Q_1, \dots, Q_s$  be the sample-specific query sets, that is,  $Q = \bigcup_{i=1}^s (\{i\} \times Q_i)$ , and let  $G$  be the contradiction graph as per Definition 15. If  $G$  is not colorable by  $m$  colors, then  $|\{x_1, \dots, x_s\}| > m$ . And if  $G$  is colorable by  $m$  colors, then there exists  $\hat{P}$  with  $|\text{supp}(\hat{P})| \leq m$  and a sequence  $y_1, \dots, y_s \in \text{supp}(\hat{P})$  such that for every  $1 \leq i \leq s$ ,  $x_i|_{Q_i} = y_i|_{Q_i}$ .*

► **Definition 18** (Explicit witness against  $m$ -support). *Let  $P$  be a distribution that is supported by a set of more than  $m$  elements. We say that  $(x_1, \dots, x_s, Q)$  is an explicit witness against  $m$ -support (of  $P$ ) if  $x_1, \dots, x_s$  are all drawn from  $P$ , and their contradiction graph contains a clique with  $m + 1$  vertices as a subgraph.*

Note that an explicit witness is in particular a witness against  $m$ -support, but the converse does not generally hold.

### 3 Quick bounds from previous results

We recall some known results for the standard model and use them to derive initial bounds on testing  $\mathcal{S}_m$ . Due to space limitation, all proofs are deferred to the full version of the paper.

Observe that, without loss of generality, we can assume that every sample is queried at least once. Using distributions over sets of vectors that are mutually 0.499-far, lower bounds for the standard model can be converted to the Huge Object model, implying in particular the following.

► **Proposition 19** (Proposition 2.8 in [9]). *Every two-sided error  $\varepsilon$ -test for  $\mathcal{S}_m$  makes at least  $\Omega(m/\log m)$  queries (for some fixed  $\varepsilon$ ).*

In the Huge Object model, different samples may be indistinguishable, hence standard-model algorithms cannot be immediately converted to Huge Object model ones. However, we can use the following reduction.

► **Lemma 20** (Theorem 2.2 in [9]). *Suppose that  $\mathcal{P}$  is testable with sample complexity  $s(n, \varepsilon)$  in the standard model, and that  $\mathcal{P}$  is closed under mapping (note that bounded support properties are closed under mapping). Then for every  $\varepsilon > 0$  there exists a non-adaptive  $\varepsilon$ -test for  $\mathcal{P}$  in the Huge Object model that uses  $3 \cdot s(m, \varepsilon)$  samples and  $O(\varepsilon^{-1} \log(\varepsilon^{-1} s(m, \varepsilon/2)))$  queries per sample.*

► **Proposition 21** (combining [14] and [9]). *There exists a two-sided  $\varepsilon$ -test for  $\mathcal{S}_m$  whose query complexity is  $O(\varepsilon^{-3} m \log \varepsilon^{-1})$ .*

In the above we used [14] rather than the more recent [15], since we needed a statement that holds for all values of  $\varepsilon$  (including those smaller than  $1/m$ ). Proposition 21 implies that for every fixed  $\varepsilon$  and variable  $m$ , there exists an  $O(m)$  non-adaptive two-sided error  $\varepsilon$ -test for  $\mathcal{S}_m$ . In this context we also note the following known bounds.

► **Theorem 22** (Corollary 2.3 in [9]). *For every  $\varepsilon > 0$  and  $m \geq 2$ , there exists a non-adaptive one-sided  $\varepsilon$ -testing algorithm for  $\mathcal{S}_m$  that takes  $O(\varepsilon^{-1} m)$  samples and makes  $O(\varepsilon^{-2} m \log(m/\varepsilon))$  queries.*

► **Theorem 23** (Theorem 6.1 in [1]). *For every  $\varepsilon > 0$  and  $m \geq 2$ , there exists an adaptive one-sided  $\varepsilon$ -testing algorithm for  $\mathcal{S}_m$  that takes  $O(\varepsilon^{-1} m)$  samples and makes  $O(\varepsilon^{-1} m^2)$  queries.*

This immediately implies an upper bound of  $O(\varepsilon^{-1} m)$  samples for  $\varepsilon$ -testing  $\mathcal{S}_m$  in the standard model of distribution testing. As can be expected, this is tight. The following proposition is considered common knowledge, but for the sake of completeness we prove it in the full version of the paper.

► **Proposition 24.** *Every one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$  takes at least  $\Omega(\varepsilon^{-1} m)$  samples in the standard model.*

As with Proposition 19, this can be converted to a Huge Object model bound.

► **Proposition 25.** *Every one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$  in the Huge Object model must make at least  $\Omega(\varepsilon^{-1}m)$  queries as well.*

In this paper we improve this proposition, showing a gap between the standard model and the Huge Object model for one-sided error tests.

## 4 Overview of our proofs

In this section we state our main results and give an overview of how to obtain them. The full proofs appear in the full version.

### 4.1 Two-sided, non-adaptive lower-bound

► **Theorem 26.** *Every non-adaptive  $\varepsilon$ -test for  $\mathcal{S}_m$  must make  $\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$  queries, even if it has two-sided error.*

We first describe our lower bound for  $\mathcal{S}_2$ , which holds the main ideas also for  $\mathcal{S}_m$ . We begin by analyzing a restricted form of non-adaptive algorithms, which we call *rectangle algorithms*. A rectangle algorithm is characterized by the number of samples  $s$  and a set  $I$  of indices. Every sample is queried at the indices of  $I$ , hence the query complexity is  $s \cdot |I|$ . We say that  $|I|$  is the “width” of the rectangle and that the number of samples is its “height”.

Consider the following  $O(\varepsilon^{-1})$ -query rectangle algorithm: for some hard-coded parameter  $\beta > 0$ , it chooses a set  $I$  of  $O(\beta^{-1})$  indices, and then it takes  $O(\beta\varepsilon^{-1})$  samples, and queries every sample on all indices of  $I$ .

Now consider the following form of inputs. For some  $\alpha > 0$  and two strings  $a$  and  $b$  for which  $d(0, a), d(0, b), d(a, b) = \Theta(\alpha)$ , let  $P$  be the following distribution. The string 0 is picked with probability  $1 - c\alpha^{-1}\varepsilon$ , the string  $a$  with probability  $\frac{c}{2} \cdot \alpha^{-1}\varepsilon$  and the string  $b$  with probability  $\frac{c}{2} \cdot \alpha^{-1}\varepsilon$ , where  $c > 1$  is some global constant.

Intuitively, the algorithm finds a witness against 2-support if there is a query common to  $a$  and  $b$ , at an index  $j$  that is not always zero (we call such  $j$  a *non-zero index*). That is, there are two necessary conditions to reject: the algorithm must get both  $a$  and  $b$  as samples, and it must query at an index  $j$  for which  $(a)_j \neq (b)_j$ .

The expected number of non-zero samples that the algorithm gets is  $O(\alpha^{-1}\beta)$ . If  $\alpha$  is much greater than  $\beta$ , then with high probability the algorithm only gets all-zero samples and cannot even distinguish the input distribution from the deterministic all-zero one.

If  $\alpha$  is much smaller than  $\beta$ , then with high probability all queries are made in “zero indices” and the algorithm again cannot even distinguish the input distribution from the deterministic all-zero one. Thus, the algorithm can reject the input with high probability only if  $\alpha \approx \beta$ .

Our construction of  $D_{\text{no}}$  chooses  $\alpha = 2^k$  where  $k$  is distributed uniformly over its relevant range, to ensure that a rectangle algorithm (with a fixed  $\beta$ ) “misses”  $\alpha$  with high probability. Intuitively, the idea is that a non-adaptive algorithm must accommodate a large portion of the possible values of  $\alpha$ , which would lead to an additional  $\log \varepsilon^{-1}$  factor. Then, we show that given an input drawn from  $D_{\text{no}}$ , if the algorithm did not distinguish two non-zero elements, then the distribution of runs looks exactly the same as the distribution of runs of the same algorithm given an input drawn from  $D_{\text{yes}}$ , which is supported over 0 and a single  $a$ .

To show that the above distributions defeat any non-adaptive algorithm (not just rectangle algorithms), we analyze every index  $1 \leq j \leq n$  according to the number of samples which are queried in that index. If few samples are queried, then this index has a high probability of not hitting two non-zero samples, rendering it useless (we gain an important advantage by noting

that querying  $j$  from at least two non-zero samples is required for it to be useful). If many samples are queried on  $j$  then this index may hit many samples, but only few indices can host many queries, which gives us a high probability of all of them together not containing a non-zero index among them.

To extend this result to  $m \geq 2$ , for every  $t \geq 2$  we define a distribution  $D_{\text{no}}^t$  over inputs that are supported by  $t + 1$  elements (one of them being the zero vector), and also  $\varepsilon$ -far from being supported by  $m$  elements (for every  $m \leq t/2 + 1$ ). As before, we define  $D_{\text{yes}}$  as a distribution over inputs supported by 2 elements, which is identical to  $D_{\text{no}}^1$ , and then we proceed with the same argument as before.

► **Definition 27** ( $D_{\text{no}}^t, D_{\text{yes}}$ ). *The distribution  $D_{\text{no}}^t$  (over a set of distributions) is obtained by the following process. Draw  $\alpha$  such that  $\log_2 \alpha^{-1}$  is uniform over  $\{2, \dots, \lfloor \log_2 \varepsilon^{-1} \rfloor - 2\}$ . Draw a set  $D \subseteq \{1, \dots, n\}$  such that for every  $1 \leq j \leq n$ ,  $\Pr[j \in D] = 4\alpha$ , independently. Then, for every  $1 \leq k \leq t$ , draw a set  $A_k \subseteq D$  such that for every  $j \in D$ ,  $\Pr[j \in A_k | j \in D] = \frac{1}{2}$ , independently. The resulting input is defined as the following distribution over  $\{0, 1\}^n$ :*

$$P : \begin{cases} 0 & \text{with probability } 1 - 2\alpha^{-1}\varepsilon \\ 1_{A_1} & \text{with probability } 2\alpha^{-1}\varepsilon/t \\ \vdots & \\ 1_{A_t} & \text{with probability } 2\alpha^{-1}\varepsilon/t \end{cases}$$

The distribution  $D_{\text{yes}}$  is identical to  $D_{\text{no}}^1$

## 4.2 One-sided, non-adaptive upper bound

► **Theorem 28.** *There exists a one sided  $\varepsilon$ -testing algorithm for  $\mathcal{S}_m$  making  $O(\varepsilon^{-1} \log \varepsilon^{-1} \cdot m \log m)$  queries.*

Let us first consider a “reverse engineering” algorithm: for every  $\ell = 2^0, 2^1, \dots, 2^{\log \varepsilon^{-1}}$ , we query  $\Theta((\varepsilon^{-1}/\ell) \cdot \log m)$  indices that are common to at least  $\ell \cdot m$  samples. Intuitively, according to the analysis of the two-sided lower bound, the algorithm should have roughly  $\Omega(m \log m)$  indices that distinguish pairs of elements, which suffice for a contradiction graph that contains an  $m + 1$ -clique.

This intuition appears to be lacking when it comes to showing the correctness of this construction for inputs that lack the special form of  $D_{\text{no}}^t$  from Definition 27. To be able to handle distance combinations (instead of just one “ $\alpha$ ” as above), we use a concept of “valid compositions”.

► **Definition 29.** *A valid composition is an ordered combination of samples  $(x_1, \dots, x_k)$  and a sequence of non-decreasing scales  $(a_2, \dots, a_k)$ , for which the distances are bounded by  $d(x_i, \{x_1, \dots, x_{i-1}\}) > 2^{-a_i-1}$ .*

Querying according to index sets whose random choice follows the prescribed distances distinguishes all elements in a composition with high probability. Our goal is to show the existence of valid compositions of  $m + 1$  elements in order to ensure that we find an explicit witness, and thus establish the upper bound. In particular, the algorithm (Algorithm 2) works as follows. It looks for a set  $A$  for of size at least  $m + 1$  whose elements are fully distinguishable using queries.

At first, the algorithm chooses  $I_0 \subseteq I_1 \subseteq \dots \subseteq I_{\log \varepsilon^{-1}} \subseteq \{1, \dots, n\}$ , where  $I_a$  consists of  $\lceil 2^{a+2} \log(m + 1) \rceil$  indices drawn uniformly and independently.

## 46:12 Support Testing in the Huge Object Model

The algorithm takes  $1 + 32\varepsilon^{-1}m$  samples. Except for the first sample, they are partitioned into  $2m$  “blocks” of at most  $16\varepsilon^{-1}$  samples each. For every  $1 \leq k \leq 2m$  and  $0 \leq a \leq \log \varepsilon^{-1}$ , the algorithm takes a sequence  $S_{a,k}$  of  $2^{3-a}\varepsilon^{-1}$  new samples, and queries every sample in it at the indices of  $I_a$ .

The algorithm rejects if there exists a *distinguishable composition* of size  $m + 1$  (which in particular is also a witness against  $\mathcal{S}_m$ ).

► **Definition 30.** *We say that a composition is a distinguishable composition if for every  $1 \leq i_1 < i_2 \leq k$  there exists a query  $j \in I_{a_{i_1}} \cap I_{a_{i_2}}$  for which  $(x_{i_1})_j \neq (x_{i_2})_j$ .*

■ **Algorithm 2** Non-adaptive construction of a valid composition.

---

```

choose indices  $i_1, \dots, i_{\lceil 4\varepsilon^{-1} \log(m+1) \rceil}$  uniformly and independently, with repetitions.
for  $0 \leq a \leq \log \varepsilon^{-1}$  do
    let  $I_a = \{i_1, \dots, i_{\lceil 2^{a+2} \log(m+1) \rceil}\}$ .
take a sample  $u$ .
query  $u$  at  $I_{\log \varepsilon^{-1}}$ .
for  $k$  from 1 to  $2m$  do
    for  $a$  from 0 to  $\log \varepsilon^{-1}$  do
        take  $2^{3-a}\varepsilon^{-1}$  new samples, denoting the sequence by  $S_{a,k}$ .
        query all samples in  $S_{a,k}$  at  $I_a$ .
if there exists a distinguishable composition of size  $m + 1$  then
    return REJECT
else
    return ACCEPT

```

---

However, it is not clear that “long” valid compositions even exist. To show their existence with high probability whenever the input is  $\varepsilon$ -far from having support size at most  $m$ , we use an extremal probabilistic argument. For this purpose, for a composition  $A$  we define its rank to be its scale sequence  $\vec{r}(A) = (a_2, \dots, a_k)$ , and refer to the lexicographic order over ranks (in particular considering a proper prefix of a sequence to be smaller in that order).

We then show that if the input is  $\varepsilon$ -far from having support size  $m$ , then with high probability no composition with at most  $m$  elements has maximal rank. This implies that the maximally ranked composition cannot have less than  $m + 1$  elements, leading with high probability to finding an explicit witness against  $m$ -support through the queries made to this composition.

To show the above in the full version, for every  $K \subseteq \{1, \dots, 2m\}$  we define the event that the blocks indexed by  $K$  are exactly those that contain the maximally ranked composition. We then show that if the length of this composition is at most  $m$ , (and the input is  $\varepsilon$ -far from the property), then the probability of this event happening is small enough to deploy a union bound argument against all such events.

### 4.3 One-sided, adaptive upper bound

► **Theorem 31.** *There exists a one-sided  $\varepsilon$ -testing algorithm for  $\mathcal{S}_m$  making  $O(\varepsilon^{-1}m \log m \cdot \min\{\log \varepsilon^{-1}, \log m\})$  queries.*

We adaptively construct a distinguishing sequence that resembles a valid composition (see Definition 29), but at some point we decide to “give up” and change phase to another way of querying that is more efficient under some conditions. Luckily, the condition that makes us give up implies them. For every distance scale, from  $\Omega(1)$  to  $\frac{1}{m}$ , we use the “fishing expedition” paradigm (Lemma 14) using Algorithm 3 as the subroutine  $\mathcal{A}$ , to extend our sequence with as many elements as we can until we are certain enough that it is no longer effective to look for them (or until we find a witness against  $m$ -support). This phase is described in Algorithm 4.

■ **Algorithm 3** Adaptive one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$ , a single batch.

---

**parameters**  $\varepsilon > 0$ ,  $A$ ,  $m \geq 2$ ,  $0 \leq a \leq \lceil \log m \rceil$  where  $|A| \leq m$ .  
**input** A distribution  $P$ .  
**choose** a set  $J$  of  $\lceil 2^{a+2} \log m \rceil$  indices uniformly and independently.  
**query**  $X$  at  $J$  for every  $X \in A$ .  
**take**  $\lceil 2^{2-a} \varepsilon^{-1} \log m \rceil$  samples.  
**query** each new sample at  $J$ .  
**if** there exists a sample  $Y$  for which  $Y|_J \neq X_J$  for every  $X \in A$  **then**  
    **set**  $A \leftarrow A \cup \{Y\}$ .  
    **return** SUCCESS with  $(Y, J)$ .  
**else**  
    **return** FAIL

---

■ **Algorithm 4** Adaptive one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$ , first phase.

---

**parameters**  $\varepsilon > 0$ ,  $m \geq 2$ .  
**input** A distribution  $P$ , a set  $A \subseteq \text{supp}(P)$  of distinguishable elements.  
**for**  $a$  **from** 0 **to**  $\lceil \log m \rceil$  **do**  
    **let**  $k_a = m + 1 - |A|$ .  
    **run** Algorithm 1 (“fishing expedition”) with parameters  $k = k_a$ ,  $q = \frac{1}{4^{\lceil \log m + 1 \rceil}}$ ,  $p = \frac{1}{3}$ ,  
    and  $\mathcal{A} =$  Algorithm 3 (a single batch).  
    **if**  $|A| \geq m + 1$  **then**  
        **return** REJECT  
    Proceed to the second phase with  $A$ .

---

Unfortunately, it is possible that at some point the algorithm is certain enough that it is no longer effective to look for elements in any of these scales. At this point, we observe that the contribution of elements with small distance scale to the distance of the input from  $\mathcal{S}_m$  is still  $\Omega(\varepsilon)$  (that is, we can safely ignore the “rare large-distance elements”). To make use of this observation, the algorithm shifts to the second phase, looking for elements with small distances in a way which does not follow the theme of looking for valid compositions.

In the small distance scale phase we construct and maintain a “decision tree” data structure over the existing elements, so that for every element that we need to compare to the existing elements, we can rule out in advance, using only  $O(m)$  many queries, all but one of them. This allows us to save queries, since the smaller distances require the querying of relatively many indices for a comparison, which would have been very inefficient to perform for all existing elements. See Algorithm 5 for precise details.

■ **Algorithm 5** Adaptive one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$ , a single iteration of the second phase.

---

**input** A sample  $Y \in \text{supp}(P)$ ,  $A \subseteq \text{supp}(P)$ , a decision tree  $\mathcal{T}$ ;  $|A| \geq 1$ .  
**invariant**  $\mathcal{T}$  has  $|A|$  leaves corresponding to  $A$ 's elements.  
**choose** a set  $J$  of  $m$  indices uniformly, independently, with repetitions.  
**let**  $X \in A$  for which  $\mathcal{T}(Y) = \mathcal{T}(X)$  (using up to  $|A|$  queries to  $Y$  to follow  $\mathcal{T}$  and find  $X$ ).  
**query**  $X, Y$  at  $J$ .  
**if**  $Y|_J \neq X|_J$  **then**  
    **set**  $A \leftarrow A \cup \{Y\}$ .  
    **add**  $Y$  to  $\mathcal{T}$  (using a distinguishing index  $j \in J$  to split the leaf of  $X$ ).

---

Finally, we combine the above procedure to obtain our desired algorithm:

■ **Algorithm 6** Adaptive one-sided  $\varepsilon$ -test for  $\mathcal{S}_m$ .

---

**input** A distribution  $P$ .  
**if**  $\varepsilon \geq \frac{1}{m^2}$  **then**  
    **run** Algorithm 2 and **return** its answer.  
**take** the first sample  $u$ .  
**set**  $A \leftarrow \{u\}$ .  
**run** Algorithm 4 (possibly modifying  $A$ , possibly rejecting).  
**construct** a decision tree  $\mathcal{T}$  based on  $A$ .  
**invariant**  $\mathcal{T}$  has  $|A|$  leaves corresponding to  $A$ 's elements.  
**for**  $\lceil 48\varepsilon^{-1} \rceil$  **times do**  
    **draw** another sample  $Y$ .  
    **run** Algorithm 5 with  $(Y, A, \mathcal{T})$  (note that  $A, \mathcal{T}$  may have been modified).  
    **if**  $|A| \geq m + 1$  **then**  
        **return** REJECT  
**return** ACCEPT

---

#### 4.4 One-sided lower-bounds

► **Theorem 32.** *Every one-sided (possibly adaptive)  $\varepsilon$ -test for  $\mathcal{S}_m$  must make  $\Omega(\varepsilon^{-1}m \log \varepsilon^{-1})$  queries.*

We prove that an algorithm obtains a witness against  $m$ -support if and only if the contradiction graph (Definition 15) is not  $m$ -colorable. Hence we look for the lower bound on the number of queries needed to construct a non- $m$ -colorable contradiction graph.

We observe that, given a query set, every index  $j$  describes a biclique contradiction graph whose classes are “all samples queried at  $j$  for which  $x_j = 0$ ” and “all samples queried at  $j$  for which  $x_j = 1$ ”. The contradiction graph is the union of these graphs. Specifically, we define the notion of *capacity*.

► **Definition 33** (Capacity of an edge cover). *Let  $G$  be a graph over a set  $V$  vertices and let  $\mathcal{G} = (G_1, \dots, G_k)$  be a sequence of graphs over  $V_1, \dots, V_k \subseteq V$  such that  $G = \bigcup_{i=1}^k G_i$ . We define the capacity of  $\mathcal{G}$  as  $\text{cap}(\mathcal{G}) = \sum_{i=1}^k |V_k|$ .*

The following observation follows directly from the definition of capacity.

► **Observation 34.** *Let  $P$  be a distribution over  $\{0, 1\}^n$ ,  $x_1, \dots, x_s \in \text{supp}(P)$  be a set of samples and  $Q \subseteq \{1, \dots, s\} \times \{1, \dots, n\}$  be a query set. Let  $S_1, \dots, S_n$  be the index-specific query sets, that is,  $Q = \bigcup_{j=1}^n (S_j \times \{j\})$ . In other words, for every  $j$ , all samples in  $S_j$*



are queried at the index  $j$ . Let  $\mathcal{G} = (G_1, \dots, G_n)$  be the edge cover of the contradiction graph (Definition 15) implied by  $(x_1, \dots, x_s; Q)$ : for every  $1 \leq j \leq n$ ,  $G_j$  is the complete bipartite graph whose vertices are  $S_j$  and the sides are  $L_j = \{i \in S_j | (x_i)_j = 0\}$  and  $R_j = \{i \in S_j | (x_i)_j = 1\}$ . In this setting,  $\text{cap}(\mathcal{G}) = |Q|$ .

The following lemma is crucial for our one-sided testing lower bounds.

► **Lemma 35** ([10, 11, 2]). *Let  $V$  be a set of vertices, and let  $\mathcal{G} = (G_1, \dots, G_k)$  be an edge cover of the  $V$ -clique such that all graphs  $G_1, \dots, G_k$  are bipartite. Then  $\text{cap}(\mathcal{G}) \geq |V| \log_2 |V|$ .*

It can be extended to any non- $m$ -colorable graph, which is what we need.

► **Lemma 36.** *Let  $G$  be a graph over a set  $V$  of vertices that is not  $m$ -colorable, and let  $\mathcal{G} = (G_1, \dots, G_k)$  be an edge cover of  $G$  such that all graphs  $G_1, \dots, G_k$  are bipartite. Then  $\text{cap}(\mathcal{G}) \geq (m+1) \log_2(m+1)$ .*

Then we extend our analysis in two ways, one of which applies to non-adaptive algorithms (giving a  $\log \varepsilon^{-1}$  factor) and the other also applies to adaptive ones (giving a  $\log m$  factor).

For non-adaptive algorithms, we extend the analysis of the two-sided bound to show that a one-sided algorithm for  $\mathcal{S}_m$  requires  $\Omega(\varepsilon^{-1} m \log \varepsilon^{-1})$  many queries. The following shows the hardness of “gathering a witness against  $\mathcal{S}_m$ ”, which allows for a more versatile argument as compared to the indistinguishability argument that we use for the lower bound of Theorem 26.

We use  $D_{\text{no}}^t$  (Definition 27) using  $t = 4m/3$ . For a non-adaptive algorithm that makes less than  $O(\varepsilon^{-1} m \log \varepsilon^{-1})$  queries, the probability that it distinguishes two specific non-zero elements is  $\frac{1}{16}$ . Considering the contradiction graph, excluding the vertex corresponding to the zero vector, we show that the expected number of edges is at most  $\frac{1}{16} \binom{t}{2}$ . By Markov’s inequality, with probability higher than  $\frac{2}{3}$ , there are less than  $\binom{3t/4-1}{2} = \binom{m-1}{2}$  edges, meaning that this subgraph is colorable using  $m-1$  colors. Combined with the vertex corresponding to the zero vector, the contradiction graph is colorable by  $m$  colors, hence it cannot be a witness against being supported on only  $m$ -support.

For the other bounds we use Lemma 36. To show a lower bound against non-adaptive algorithms, we construct a distribution in which a single, “anchor” element is drawn with probability  $1 - \Theta(\varepsilon)$ . This way, for every non-adaptive algorithm that makes only  $o(\varepsilon^{-1} m \log m)$  many queries, the expected number of queries applied to other elements is  $o(m \log m)$ . By Markov’s inequality, with probability  $\frac{2}{3}$ , only  $o(m \log m)$  queries are made in non-zero elements, and in this case, there cannot be a witness against  $m-1$  other elements.

This construction cannot be immediately applied to adaptive algorithms, since they can use adaptivity to avoid wasting queries on the anchor element. To overcome this issue, we use two additional methods. The first one is using very short strings, that is, we focus on distributions over  $\{0, 1\}^{O(\log m)}$  that are  $\varepsilon$ -far from having  $m$  elements in their support (later we prove that the bound also holds for arbitrarily large  $n$  using a simple repetition technique). The second method involves using shared-secret code ensembles [5] that guarantee, in an appropriate setting, that if the algorithm makes less than  $O(\log m)$  queries in an individual sample, then it gathers no information at all. This way, for every individual sample, the algorithm either behaves similarly to a non-adaptive algorithm or makes at least a fixed portion of the maximum number of queries. The exact argument requires a careful analysis of the decision tree of the algorithm.

---

**References**

---

- 1 Tomer Adar and Eldar Fischer. Refining the adaptivity notion in the huge object model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London, United Kingdom*, volume 317. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- 2 Noga Alon. On bipartite coverings of graphs and multigraphs. *arXiv preprint*, 2023. [arXiv:2307.16784](https://arxiv.org/abs/2307.16784).
- 3 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001.
- 4 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE, 2000.
- 5 Omri Ben-Eliezer, Eldar Fischer, Amit Levi, and Ron D Rothblum. Hard properties with (very) short pcpps and their applications. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, 2020.
- 6 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.
- 7 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.
- 8 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75, 2011.
- 9 Oded Goldreich and Dana Ron. Testing distributions of huge objects. *TheoretCS*, 2, 2023.
- 10 Georges Hansel. Nombre minimal de contacts de fermeture nécessaires pour réaliser une fonction booléenne symétrique de  $n$  variables. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES*, 258(25):6037, 1964.
- 11 Gyula Katona and Endre Szemerédi. On a problem of graph theory. *Studia Scientiarum Mathematicarum Hungarica*, 2:2328, 1967.
- 12 Ronitt Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.
- 13 Gregory Valiant and Paul Valiant. Estimating the unseen: an  $n/\log(n)$ -sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 685–694, 2011.
- 14 Gregory Valiant and Paul Valiant. Estimating the unseen: improved estimators for entropy and other properties. *Journal of the ACM (JACM)*, 64(6):1–41, 2017.
- 15 Yihong Wu and Pengkun Yang. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *The Annals of Statistics*, 47(2):857–883, 2019.
- 16 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

# Upper Bounds on the 2-Colorability Threshold of Random $d$ -Regular $k$ -Uniform Hypergraphs for $k \geq 3$

Evan Chang ✉

Massachusetts Institute of Technology, USA

Neel Kolhe ✉

University of California, Berkeley, USA

Youngtak Sohn ✉ 

Department of Mathematics, Massachusetts Institute of Technology, USA

---

## Abstract

For a large class of random constraint satisfaction problems (CSP), deep but non-rigorous theory from statistical physics predict the location of the sharp satisfiability transition. The works of Ding, Sly, Sun (2014, 2016) and Coja-Oghlan, Panagiotou (2014) established the satisfiability threshold for random regular  $k$ -NAE-SAT, random  $k$ -SAT, and random regular  $k$ -SAT for large enough  $k \geq k_0$  where  $k_0$  is a large non-explicit constant. Establishing the same for small values of  $k \geq 3$  remains an important open problem in the study of random CSPs.

In this work, we study two closely related models of random CSPs, namely the 2-coloring on random  $d$ -regular  $k$ -uniform hypergraphs and the random  $d$ -regular  $k$ -NAE-SAT model. For every  $k \geq 3$ , we prove that there is an explicit  $d_*(k)$  which gives a satisfiability upper bound for both of the models. Our upper bound  $d_*(k)$  for  $k \geq 3$  matches the prediction from statistical physics for the hypergraph 2-coloring by Dall'Asta, Ramezanzpour, Zecchina (2008), thus conjectured to be sharp. Moreover,  $d_*(k)$  coincides with the satisfiability threshold of random regular  $k$ -NAE-SAT for large enough  $k \geq k_0$  by Ding, Sly, Sun (2014).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures; Mathematics of computing  $\rightarrow$  Random graphs

**Keywords and phrases** Random constraint satisfaction problem, replica symmetry breaking, interpolation bound

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.47

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2308.02075> [14]

**Funding** *Youngtak Sohn*: Supported by Simons-NSF Collaboration on Deep Learning NSF DMS-2031883 and Elchanan Mossel's Vannevar Bush Faculty Fellowship award ONR-N00014-20-1-2826.

**Acknowledgements** We thank the MIT PRIMES program and its organizers Pavel Etingof, Slava Gerovitch, and Tanya Khovanova for making this possible. Y.S. thanks Elchanan Mossel, Allan Sly, and Nike Sun for encouraging feedbacks.

## 1 Introduction

In this work, we study the 2-coloring on random  $d$ -regular  $k$ -uniform hypergraphs and the random  $d$ -regular  $k$ -NAE-SAT model for  $k \geq 3$ . We establish an explicit well-defined upper bound on the satisfiability/colorability threshold that holds for every  $k \geq 3$ , which is conjectured to be sharp in statistical physics [25] for hypergraph 2-coloring, and matches the previous rigorous results for random regular  $k$ -NAE-SAT model for  $k$  large enough [28].



© Evan Chang, Neel Kolhe, and Youngtak Sohn;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 47; pp. 47:1–47:23



Leibniz International Proceedings in Informatics

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

Given a  $k$ -uniform hypergraph with  $n$  nodes and  $m$  hyperedges, where every edge consists of  $k$  nodes, a hypergraph 2-coloring is an assignment of colors from  $\{\text{red}, \text{blue}\} \equiv \{0, 1\}$  to the nodes such that there is no monochromatic hyperedge. If there is such a 2-coloring, the hypergraph is said to be colorable or satisfiable. It is a typical example of a constraint satisfaction problem (CSP) that has been studied extensively in combinatorics and computer science literature [54, 7, 2, 24, 34, 38, 39].

A  $k$ -NAE-SAT problem is another closely related CSP studied in computer science [19, 28, 57, 50, 56], which can be viewed as a variant of the infamous  $k$ -SAT problem [41]. A  $k$ -SAT formula is a boolean CNF formula with  $n$  variables formed by taking the AND of  $m$  clauses, which is the OR of  $k$  variables or their negations. Then, a NAE-SAT solution  $\underline{x} \in \{0, 1\}^n$  is an assignment such that  $\underline{x}$  and its negation  $\neg \underline{x}$  evaluates true in the formula. Thus, denoting each clause as a hyperedge, if no variable is negated in every clause, then a NAE-SAT solution is equivalent to a hypergraph 2-coloring.

A significant direction of research on satisfiability has involved examining the large-system limit of randomly generated problem instances. The study of random constraint satisfaction problems (CSPs) aims to discern typical behaviors and phase transitions in these systems as the number of variables  $n$  and the number of constraints  $m$  tends to infinity with a fixed ratio  $\alpha \equiv \frac{m}{n}$ . In this sparse regime, there has been considerable effort into identifying the satisfiability transition, or the critical density, denoted by  $\alpha_{\text{sat}}$ , beyond which solutions cease to exist [6, 5, 3, 23].

Many of the sparse CSPs belong to a broad universality class called the one-step-replica-symmetry-breaking (1RSB) class from statistical physics [43] (see Chapter 19 of [47] for a survey) - including 2-coloring on random regular  $k$ -uniform hypergraphs, random regular  $k$ -NAE-SAT, and random  $k$ -SAT for  $k \geq 3$ . The 1RSB class refers to CSPs which are predicted to possess a single layer of hierarchy of well-separated clusters, where a cluster roughly refers to a dense region of the solution space. A shared characteristic of these problems is that in a non-trivial regime below  $\alpha_{\text{sat}} \equiv \alpha_{\text{sat}}(k)$ , the number of solutions fails to concentrate about its mean due to the clustering effect. This effect thus prevents standard first and second moment methods from locating the exact transition, presenting a significant mathematical challenge.

Despite such difficulties, breakthroughs were made to successfully locate the satisfiability threshold of the random regular  $k$ -NAE-SAT [31], the random  $k$ -SAT [32], and random regular  $k$ -SAT [21] for large enough  $k \geq k_0$ , where  $k_0$  is a non-explicit large absolute constant. These works carried out a demanding second moment method to the number of clusters instead of the number of solutions based on intuitions from statistical physics [46] and previous mathematical works [6, 20, 21]. See Section 1.1 for further literature.

However, for small values of  $k \geq 3$ , locating the satisfiability threshold for CSPs in the 1RSB class remains an important open problem. Indeed, for all the aforementioned models in 1RSB class, the physicists conjecture an explicit value  $\alpha_*(k)$  for  $\alpha_{\text{sat}}(k)$ , the 1RSB threshold, which is expected to be correct for all  $k \geq 3$  [45, 46, 25]. The methods of [31, 32, 21] crucially uses the fact that  $k$  is large enough for their second moment method to succeed.

In this work, we consider 2-coloring on random  $d$ -regular  $k$ -uniform hypergraphs, where the random hypergraph is generated uniformly at random from the set of  $k$ -uniform hypergraphs such that every variable participates in exactly  $d$  hyperedges. We also consider random  $d$ -regular NAE-SAT, where  $k$ -SAT formula is generated uniformly at random with the condition that every variable participates in exactly  $d$  clauses. We establish an upper bound  $d_*(k)$  on the satisfiability thresholds for these problems for every  $k \geq 3$ , which is sharp [28] for random regular  $k$ -NAE-SAT for large  $k \geq k_0$  and conjectured to be sharp [25] for  $k \geq 3$  for hypergraph 2-coloring.

► **Theorem 1.1.** For  $k \geq 3$  and  $d_{\text{lb}}(k) \leq d \leq d_{\text{ub}}(k)$ , where  $d_{\text{lb}}(k), d_{\text{ub}}(k)$  are defined in (1.4) below, there exists a unique solution  $x_\star \equiv x_\star(k, d)$  to the equation

$$d = 1 + \left( \log \frac{1-2x}{1-x} \right) / \log \left( \frac{1-2x^{k-1}}{1-x^{k-1}} \right) \quad \text{on the interval} \quad \frac{1}{2} - \frac{1}{2^k} \leq x \leq \frac{1}{2}. \quad (1.1)$$

Define  $d_\star(k)$  by the largest zero of the explicit function

$$\star\Phi(d) := -\log(1-x) - d(1-k^{-1} - d^{-1}) \log(1-2x^k) + (d-1) \log(1-x^{k-1}), \quad (1.2)$$

where the existence of the root of  $\star\Phi(d)$  is guaranteed in the interval  $[d_{\text{lb}}(k), d_{\text{ub}}(k)]$ .

Then, for  $k \geq 3$ , and  $d > d_\star(k)$ , the random  $d$ -regular  $k$ -uniform hypergraph is not 2-colorable with probability tending to one as the graph size  $n \rightarrow \infty$ . Similarly for  $k \geq 3$  and  $d > d_\star(k)$ , then the random  $d$ -regular  $k$ -NAE-SAT instance is not satisfiable with probability tending to one as  $n \rightarrow \infty$ .

A matching lower bound was obtained in [28] for large enough  $k \geq k_0$  in random  $d$ -regular NAE-SAT by a demanding second moment method. Our proof is based on an interpolation method from statistical physics [35, 37, 52]. We give a proof outline in Section 1.2.

We emphasize that for any  $k \geq 3$ , determining the colorability threshold for 2-coloring on random  $d$ -regular  $k$ -uniform hypergraphs was previously open, thus Theorem 1.1 for 2-coloring is novel even for large  $k$ . Although it is expected that the colorability threshold for the model matches the satisfiability threshold for random regular  $k$ -NAE-SAT, it is highly non-trivial to modify the proof techniques for random regular NAE-SAT [31] to the 2-coloring model since many of the arguments in [31] crucially take advantage of the randomness of clauses. For example, any  $\underline{x} \in \{0, 1\}^n$  has the same probability of being a NAE-SAT solution by the randomness of the clauses while this is obviously not true for the 2-coloring model. As we see below, even the calculation of the first moment of the solutions is substantially more involved for the 2-coloring model. Let  $Z_{\text{NAE}}$  be the number of solutions of random  $d$ -regular  $k$ -NAE-SAT, then it is trivial to calculate  $\mathbb{E}Z_{\text{NAE}}$  exactly by taking advantage of the randomness of the clauses:

$$\mathbb{E}Z_{\text{NAE}} = 2^n (1 - 2^{-k+1})^m = \exp \left( n \left( \log 2 + \alpha \log (1 - 2^{-k+1}) \right) \right) =: \exp(n\Phi_k(\alpha)). \quad (1.3)$$

On the other hand, if we denote  $Z_{\text{COL}}$  by the number of 2-colorings on random  $d$ -regular  $k$ -uniform graphs, then estimating  $\mathbb{E}Z_{\text{COL}}$  is more delicate: we appeal to the idea of exponential tilting from large deviations theory [26] and local central limit theorem [13] to prove that  $\mathbb{E}Z_{\text{COL}}$  is of the same order as  $\exp(n\Phi_k(\alpha))$  in Lemma 1.7 below. Using the interpolation bound which is simpler than moment calculations, we clarify a simple mechanism (cf. Lemma 2.2) behind the identical satisfiability upper bounds for both models.

The solution  $x_\star(k, d)$  to the equation (1.1) has a mathematical interpretation. Namely,  $2x_\star(k, d)$  is the fraction of the so-called *frozen* variables in the *cluster* model. The solution  $x_\star(k, d)$  is called the *Belief Propagation* (BP) fixed point for the cluster model in statistical physics. We emphasize that addressing the uniqueness of the BP fixed point is a well-known major obstacle for many combinatorial optimization and statistical inference problems that exhibit sharp phase transitions (e.g. for spherical perceptron model [55]; see [59, Chapter 3] for a further discussion). We establish the uniqueness of the BP fixed point by showing that the *Belief Propagation recursion* (cf. (1.12)) is a contraction for  $k \geq 3$  and  $[d_{\text{lb}}(k), d_{\text{ub}}(k)]$ , which might be also useful in obtaining a matching lower bound to Theorem 1.1.

■ **Table 1** A comparison with the upper bound  $d_*(k)$  in Theorem 1.1 with the first moment threshold  $d_1(k) := \frac{k \log 2}{-\log(1-2^{-k+1})}$  for small values of  $k$ . For  $3 \leq k \leq 10$ , the values also appear in Table 1 of [25].

| k                      | 3 | 4  | 5  | 6   | 7   | 8   | 9    | 10   | 11   | 12    | 13    | 14    | 15     |
|------------------------|---|----|----|-----|-----|-----|------|------|------|-------|-------|-------|--------|
| $\lceil d_*(k) \rceil$ | 7 | 20 | 53 | 130 | 307 | 705 | 1592 | 3543 | 7802 | 17028 | 36902 | 79488 | 170340 |
| $\lceil d_1(k) \rceil$ | 8 | 21 | 54 | 131 | 309 | 708 | 1594 | 3546 | 7804 | 17031 | 36905 | 79491 | 170343 |

Since  $\mathbb{E}Z_{\text{NAE}}$  and  $\mathbb{E}Z_{\text{COL}}$  are given by  $\exp(n\Phi_k(\alpha))$  up to a constant (cf. (1.3) and Lemma 1.7), the first moment thresholds for both of the models are given by  $d_1(k) := \frac{k \log 2}{-\log(1-2^{-k+1})}$ . In Table 1, we report  $\lceil d_*(k) \rceil$  and  $\lceil d_1(k) \rceil$  for  $3 \leq k \leq 15$ . For every  $3 \leq k \leq 15$ , the upper bound  $\lceil d_*(k) \rceil$  in Theorem 1.1 improves over the first moment threshold. For large values of  $k$ ,  $d_*(k)$  improves over  $d_1(k)$  by  $\Omega(k)$  (see (1.5) below). The quantities  $d_{\text{ibd}}(k)$ , and  $d_{\text{ubd}}(k)$  are defined by

$$d_{\text{ibd}}(k) = \begin{cases} 6.74 & k = 3, \\ 16.7 & k = 4, \\ (2^{k-1} - 2)k \log 2 & k \geq 5. \end{cases} \quad d_{\text{ubd}}(k) = \begin{cases} 7.5 & k = 3, \\ 2^{k-1}k \log 2 & k \geq 4. \end{cases} \quad (1.4)$$

► **Remark 1.1.** For  $d \leq d_{\text{ibd}}(k)$  and large  $k \geq k_0$ , the second moment method applied to  $Z_{\text{NAE}}$  succeeds in showing the satisfiability for the random  $d$ -regular  $k$ -NAE-SAT model (see [28, Section 2.1]). For  $k \in \{3, 4\}$ ,  $d_{\text{ibd}}(k)$  must be adjusted to be higher to guarantee that  $\star\Phi(d)$  is well-defined, i.e. there exists a unique solution to (1.1). The value  $d_{\text{ubd}}(k) \equiv 2^{k-1}k \log 2 > d_1(k)$  for  $k \geq 4$  is a convenient upper bound for satisfiability. For  $k = 3$ , we take  $d_{\text{ubd}}(3)$  to be  $7.5 > \frac{3 \log 2}{-\log(3/4)} = d_1(3)$ , which does not change  $d_*(3)$ , but is more convenient for the proof.

Finally, we note that the large  $k$  asymptotics of  $d_*(k)$  was proven in [58, Appendix B]:

$$\alpha_*(k) \equiv \frac{d_*(k)}{k} = \left( 2^{k-1} - \frac{1}{2} - \frac{1}{4 \log 2} \right) \log 2 + o_k(1), \quad (1.5)$$

where  $o_k(1)$  denotes an error tending to zero as  $k \rightarrow \infty$ . Since  $d_1(k) = (2^{k-1} - 1/2)k \log 2 + o_k(1)$ , we have that  $d_*(k) \leq d_1(k) - \Omega(k)$ .

## 1.1 Related work

Many of the earlier mathematical works on CSPs focused on determining their satisfiability thresholds and verifying the sharpness of SAT-UNSAT transitions. For models that are known not to exhibit RSB, such goals were established. These models include random 2-SAT [15, 12], random 1-IN- $k$ -SAT [1],  $k$ -XOR-SAT [33, 27, 53], and random linear equations [8]. On the other hand, for the models which are predicted to belong to 1RSB class, intensive studies have been conducted to estimate their satisfiability threshold, as shown in [42, 6, 21] (random  $k$ -SAT), [3, 24, 19] (random  $k$ -NAE-SAT), and [4, 16, 23, 17] (random graph coloring).

More recently, the satisfiability thresholds for rCSPs that exhibits RSB have been rigorously determined for several models, namely the random regular  $k$ -NAE-SAT [31], maximum independent set on  $d$ -regular graphs [30], random regular  $k$ -SAT [21] and random  $k$ -SAT [32] for large  $k$  and  $d$ . Although determining the location of  $q$ -colorability threshold for the sparse Erdős Rényi graph is left open, the *condensation threshold*  $\alpha_{\text{cond}}$  for random graph coloring, where the *free energy* becomes non-analytic, was settled in [11]. They carried out a

technically challenging analysis based on a clever “planting” technique, where the results were further generalized to other models in [18]. Similarly, [10] identified the condensation threshold for random regular  $k$ -SAT, where each variable appears  $d/2$ -times positive and  $d/2$ -times negative. Further, in the condensation regime  $\alpha \in (\alpha_{\text{cond}}, \alpha_{\text{sat}})$ , many quantities of interest were established for random regular  $k$ -NAE-SAT with large enough  $k$ , matching the statistical physics prediction. Namely, the number of solutions at exponential scale (free energy) [58], the concentration of the *overlap* [49, 51], and the local weak limit [56] were established. Establishing the same quantities for other models in the condensation regime is still open.

The closest result to ours in the literature is by Ayre, Coja-Oghlan, and Greenhill [9], where they lower bound the chromatic number (or equivalently, upper bound the colorability threshold) of the random regular graph of any degree, which is conjectured to be tight. [9] also considers the sparse Erdős-Rényi graph, which is more complicated since the conjectured chromatic number is defined in terms of a distributional (rather than real-valued) optimization due to the randomness of the local neighborhoods. In this work, we do not consider Erdős-Rényi type problems, but we additionally address the question of the uniqueness of the BP fixed point for any  $k \geq 3$  (unique solution to the equation (1.1)). As in [9], we use an interpolation bound, which gives an upper bound of the satisfiability threshold also for the (non-regular) random  $k$ -NAE-SAT model. It would be interesting to address the uniqueness of the BP fixed point for random  $k$ -NAE-SAT and random  $k$ -sat for small  $k \geq 3$ . We refer to [55, 48, 60, 36] which addresses the uniqueness of BP fixed point for various models.

## 1.2 Proof methods

We aim to rigorously establish the upper bound for the satisfiability threshold predicted by the so-called “1RSB cavity method” from statistical physics [25]. To do so, instead of using moment methods, we use a theorem derived from the so called “interpolation method” from the theory of spin glasses developed by [35, 37, 52]. The interpolation method has been successful in upperbounding the satisfiability threshold for random  $k$ -SAT [29] for large  $k$ , the free energy for random regular  $k$ -NAE-SAT [57], and the colorability threshold for random graphs [9].

We first introduce the notations and mathematical framework that we use throughout the paper. For both the  $d$ -regular  $k$ -uniform hypergraphs and the  $k$ -NAE-SAT formula, we can represent them as (labelled)  $(d, k)$ -regular bipartite graph. Let  $V = \{v_1, \dots, v_n\}$  be the set of variables or nodes and  $F = \{a_1, \dots, a_m\}$  be the set of clauses or hyperedges. An edge is formed if the variable or node  $v_i$  is included in the clause or hyperedge  $a_j$ . For an edge  $e$ , we denote  $v(e)$  (resp.  $a(e)$ ) by the variable (resp. clause) adjacent to it.

Denote  $G = (V, F, E)$  by the resulting bipartite graph. Each variable  $v \in V$  has incident half-edges  $\delta v$ , while each clause  $a \in F$  has incident half-edges  $\delta a$ . Throughout, we denote  $\alpha \equiv \frac{m}{n} = \frac{d}{k}$ . For the NAE-SAT formula, there is an extra label for each edge  $e \in E$ , namely the *literal*  $L_e \in \{0, 1\}$ , which specifies how the variable  $v(e)$  participates in the clause  $a(e)$ . Then, the labelled graph  $\mathcal{G} = (V, F, E, \underline{L}) \equiv (V, F, E, (L_e)_{e \in E})$  represents a NAE-SAT instance.

► **Definition 1.2.** Given a NAE-SAT instance  $\mathcal{G} = (V, F, E, \underline{L})$ ,  $\underline{x} \in \{0, 1\}^V$  is a (NAE-SAT) **solution** if

$$\prod_{a \in F} \varphi((x_{v(e)} \oplus L_e)_{e \in \delta a}) = 1,$$

where for  $\underline{z} = (z_i)_{i \leq k} \in \{0, 1\}^k$ ,  $\varphi(\underline{z}) \equiv \mathbb{1}(z_1 = \dots = z_k)$ , and  $\oplus$  denotes addition mod 2. Given a graph  $G = (V, F, E)$ ,  $\underline{x} \in \{0, 1\}^V$  is a (hypergraph 2-) **coloring** if  $\underline{x}$  is a NAE-SAT solution on  $G$  with literals identically zero  $(G, \underline{0})$ .

The configuration model can be described as follows. Add  $d$  (resp.  $k$ ) half-edges adjacent to each variable (resp. each clause) so that there are total  $nd = mk$  number of half-edges adjacent to variables (resp. clauses). Thus,  $E$  can be regarded as a perfect matching between to the set of half-edges adjacent to variables to those adjacent to clauses, and hence a permutation in  $S_{nd}$ . Then, the configuration model  $\mathbf{G} = (V, F, \mathbf{E})$  is defined by taking  $\mathbf{E} \sim \text{Unif}(S_{nd})$ . For a random  $d$ -regular  $k$ -NAE-SAT instance  $\mathcal{G} = (\mathbf{G}, \mathbf{L})$ , we take the literals  $\mathbf{L} \equiv (\mathbf{L}_e)_{e \in E} \stackrel{i.i.d.}{\sim} \text{Unif}(\{0, 1\})$ .

Note that the configuration model  $\mathbf{G}$  may induce multi-edges. However, if we denote  $\mathcal{S}$  to be the event that  $\mathbf{G}$  is simple, then it is well-known that  $\mathbb{P}(\mathbf{G} \in \mathcal{S}) = \Omega(1)$  (see e.g. Chapter 9 of [40]). Thus, the configuration model is mutually contiguous with respect to the uniform distribution among all  $(d, k)$ -regular graphs, so to prove Theorem 1.1, it suffices to work with the configuration model.

In order to use the interpolation method, we consider the *positive temperature* analogs of the 2-coloring or the NAE-SAT model, which have more desirable properties due to the softness of the constraints - e.g. the concentration of the free energy as seen in Lemma 1.3 below. We introduce notations that allow us to set up the positive temperature models. Let  $S$  be a finite set and  $\underline{b} \equiv (b_s)_{s \in S}$  be a vector with  $b_s \geq 0$ . Also, let  $\mathcal{X}$  be a finite set encoding the spins and denote  $\mathfrak{F}(\mathcal{X})$  by the set of functions  $\mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$ . Let  $f : S \rightarrow \mathfrak{F}(\mathcal{X})$  be a random function which may be chosen randomly according to any distribution, i.e.  $f(\cdot; s) \in \mathfrak{F}(\mathcal{X})$  is random for  $s \in S$ , and  $f_1, \dots, f_k$  be i.i.d. copies of  $f$ . Then, define the random function  $\theta : \mathcal{X}^k \rightarrow \mathbb{R}$  as follows. For  $\underline{x} = (x_1, \dots, x_k) \in \mathcal{X}^k$ , let

$$\theta(\underline{x}) = \sum_{s \in S} b_s \prod_{i=1}^k f_i(x_i; s). \quad (1.6)$$

We will consider  $S = \mathcal{X} = \{0, 1\}$  in Definition 1.4 below, but one may also consider the case  $S \neq \mathcal{X}$  in general. We assume that there exists a constant  $\varepsilon \in (0, 1)$  such that for any  $\underline{x} \in \mathcal{X}^k$ ,

$$\varepsilon \leq 1 - \theta(\underline{x}) \leq \varepsilon^{-1} \quad \text{almost surely.} \quad (1.7)$$

On a  $(d, k)$ -regular bipartite graph  $G = (V, F, E)$ , let  $(\theta_a)_{a \in F}$  be i.i.d. copies of the random function  $\theta$ , and define the (random) Gibbs measure on  $\mathcal{X}^V$  by

$$\mu_G(\underline{x}) \equiv \frac{1}{Z(G)} \prod_{a \in F} (1 - \theta_a(\underline{x}_{\delta a})),$$

where  $Z(G)$  is the normalizing constant explicitly given by

$$Z(G) \equiv \sum_{\underline{x} \in \mathcal{X}^V} \prod_{a \in F} (1 - \theta_a(\underline{x}_{\delta a})). \quad (1.8)$$

We note that the condition (1.7) on  $\theta$  guarantees that the Gibbs measure  $\mu_G$  is “finite temperature”. In particular, if we define the free energy

$$F_n \equiv \frac{1}{n} \mathbb{E} \log Z(\mathbf{G}), \quad (1.9)$$

where  $\mathbf{G}$  is drawn from the configuration model and  $\mathbb{E}$  above is over the randomness of  $\mathbf{G}$  and randomness of  $(\theta_a)_{a \in F}$ , we have the following concentration of the free energy.



► **Lemma 1.3.** *Assume that  $\theta$  satisfies (1.7) with some constant  $\varepsilon \in (0, 1)$ . Then, for any  $\delta > 0$ , there exists a constant which only depends on  $\varepsilon, \delta > 0$  such that*

$$\mathbb{P} \left( \left| \frac{1}{n} \log Z(\mathbf{G}) - F_n \right| \geq \delta \right) \leq e^{-cn}.$$

The concentration of free energy in Lemma 1.3 is standard in literature [11, 22, 9], and we provide the proof in Section 2 for completeness.

► **Definition 1.4.** (Positive temperature models) For  $\beta > 0$ , called the inverse temperature, the positive temperature NAE-SAT model  $\theta_{\text{NAE}}(\cdot) \equiv \theta_{\text{NAE}}(\cdot; \beta)$  is defined as follows. Let  $\underline{L} \equiv (\mathbf{L}_i)_{i \leq k} \stackrel{i.i.d.}{\sim} \text{Unif}(\{0, 1\})$  be a sequence of i.i.d. Bernoulli(1/2) random variables. Then for  $\underline{x} = (x_i)_{i \leq k} \in \{0, 1\}^k$ , define

$$\theta_{\text{NAE}}(\underline{x}) \equiv \theta_{\text{NAE}}(\underline{x}; \beta) \equiv (1 - e^{-\beta}) \cdot \left( \prod_{i=1}^k (\mathbf{L}_i \oplus x_i) + \prod_{i=1}^k (\mathbf{L}_i \oplus x_i \oplus 1) \right). \quad (1.10)$$

That is, in the general form (1.6), we take  $S = \mathcal{X} = \{0, 1\}$ ,  $b_i \equiv 1 - e^{-\beta}$ , and  $f(x; 0) \equiv 1 - f(x; 1) \equiv \mathbb{1}(x \oplus \mathbf{L})$  for  $\mathbf{L} \sim \text{Unif}(\{0, 1\})$ . Moreover, the positive temperature hypergraph 2-coloring model  $\theta_{\text{COL}}(\cdot) \equiv \theta_{\text{COL}}(\cdot; \beta)$  is defined by taking  $\mathbf{L}_i \equiv 0$  above:

$$\theta_{\text{COL}}(\underline{x}) \equiv \theta_{\text{COL}}(\underline{x}; \beta) \equiv (1 - e^{-\beta}) \cdot \sum_{s \in \{0, 1\}} \prod_{i=1}^k \mathbb{1}(x_i = s), \quad (1.11)$$

which is taking  $f(x; s) = \mathbb{1}(x = s)$  in (1.6).

We note that formally taking  $\beta = \infty$  and  $\theta = \theta_{\text{COL}}(\underline{x}; \beta)$ , the corresponding partition function  $Z(G)$  equals the number of 2-coloring on  $G$ . A similar statement holds for the NAE-SAT model.

By constructing a certain sequential coupling of the given factor graph  $(\mathbf{G}, (\theta)_{a \in F})$  to a set of disjoint trees so that the free energy is monotone at every step, the interpolation method [35, 37, 52] gives an upper bound on the free energy  $F_n$  as follows: for  $\zeta \in \mathcal{P}(\mathcal{P}(\mathcal{X}))$ , where  $\mathcal{P}(A)$  denotes the set of probability measures on  $A$ , and  $\lambda \in (0, 1)$ , there exists an explicit functional  $\mathcal{P}(\zeta, \lambda) \equiv \mathcal{P}_{d, k, \theta}(\zeta, \lambda)$  such that we have  $F_n \leq \inf_{\zeta, \lambda} \mathcal{P}(\zeta, \lambda) + o_n(1)$ . By taking advantage of the interpolation method applied to positive temperature models in Definition 1.4 and the concentration of the free energy in Lemma 1.3, we prove the proposition below in Section 2.

► **Proposition 1.5.** *For a given  $k \geq 3$  and  $d$ , suppose that there is a solution  $x \in [1/2 - 1/2^k, 1/2]$  to the BP equation (1.1). Further, suppose that  $\star \Phi(d)$  in (1.2) defined with such  $x$  satisfies  $\star \Phi(d) < 0$ . Then, with probability tending to one, no NAE-SAT solution exists on  $\mathcal{G}$ . Also, with probability tending to one, no 2-coloring exists on  $\mathcal{G}$ .*

Moreover, we show that  $d_\star(k)$  in Theorem 1.1 is well-defined and that the assumptions of Proposition 1.5 are meaningful. Note that the BP equation (1.1) is equivalent to  $\Psi_d(x) = x$ , where  $\Psi_d \equiv \Psi_{k, d} : [0, 1] \rightarrow [0, 1]$  is defined by  $\Psi_d \equiv \dot{\Psi} \circ \hat{\Psi}$  with

$$\dot{\Psi}(x) \equiv \dot{\Psi}_d(x) \equiv \frac{1 - x^{d-1}}{2 - x^{d-1}}, \quad \hat{\Psi}(x) \equiv \hat{\Psi}_k(x) \equiv \frac{1 - 2x^{k-1}}{1 - x^{k-1}}. \quad (1.12)$$

The function  $\dot{\Psi}(\cdot)$  is *variable BP recursion* and  $\hat{\Psi}(\cdot)$  is *clause BP recursion* (see [31, Section 3.1] for the motivation).

► **Proposition 1.6.** *For  $k \geq 3$  and  $d \in [d_{\text{lb}d}(k), d_{\text{ub}d}(k)]$ , there exists a unique root to  $\Psi_d(x) \equiv (\hat{\Psi} \circ \hat{\Psi})(x) = x$  in the interval  $x \in [1/2 - 1/2^k, 1/2]$ . Thus,  $\star\Phi(d)$  in equation (1.2) is well-defined. Furthermore,  $d \rightarrow \star\Phi(d)$  is continuous in the interval  $d \in [d_{\text{lb}d}(k), d_{\text{ub}d}(k)]$  with  $\star\Phi(d_{\text{lb}d}(k)) > 0$  and  $\star\Phi(d_{\text{ub}d}(k)) < 0$ .*

The proof of Proposition 1.6 is given in Section 3 for  $k \geq 4$ . We refer to the full version [14] for the proof of Proposition 1.6 for  $k = 3$ , which requires extra numerical estimates. Finally, we show that the first moment  $\mathbb{E}Z_{\text{COL}}$  of the number of 2-colorings on random  $d$ -regular  $k$ -uniform hypergraphs is the same with  $\mathbb{E}Z_{\text{NAE}}$  up to a constant.

► **Lemma 1.7.** *For  $k \geq 3$ , there exist constants  $C_{k,d,i}$  for  $i = 1, 2$ , which only depends on  $k, d$  such that  $\mathbb{E}Z_{\text{COL}}/\mathbb{E}Z_{\text{NAE}} \in [C_{k,d,1}, C_{k,d,2}]$*

**Proof of Theorem 1.1.** By Proposition 1.6, the function  $\star\Phi(d)$  is well-defined and has a root in the interval  $[d_{\text{lb}d}(k), d_{\text{ub}d}(k)]$ . Moreover, since  $\star\Phi(d_{\text{ub}d}(k)) < 0$  holds and  $\star\Phi(\cdot)$  is continuous, we have  $\star\Phi(d) < 0$  for  $d \in (d_{\star}(k), d_{\text{ub}d}(k)]$ . Hence, Proposition 1.5 shows that if  $d \in (d_{\star}(k), d_{\text{ub}d}(k)]$ , then the 2-coloring of random  $d$ -regular  $k$ -uniform hypergraph and random  $d$ -regular  $k$ -NAE-SAT is not satisfiable, both with probability tending to one as  $n \rightarrow \infty$ . Further, since  $\mathbb{E}Z_{\text{COL}} \asymp_{k,d} \mathbb{E}Z_{\text{NAE}} = \exp(n(\log 2 + \alpha \log(1 - 2^{-k+1})))$  by Lemma 1.7 and  $\log 2 + \alpha \log(1 - 2^{-k+1}) < 0$  holds for  $d > d_{\text{ub}d}(k)$ , the same is true for  $d > d_{\text{ub}d}(k)$  by Markov's inequality. ◀

## 2 Satisfiability upper bound by interpolation

In this section, we prove Lemma 1.3, Proposition 1.5, and Lemma 1.7. We prove Proposition 1.5 in Section 2.1 based on the interpolation bound from statistical physics [35, 37]. In Section 2.2, we prove Lemma 1.3 based on Azuma Hoeffding's inequality applied to the Doob martingale with respect to clause revealing filtration. In Section 2.3, we prove Lemma 1.7 based on the local central limit theorem.

### 2.1 Proof of Proposition 1.5

Throughout, we assume that we are given  $k \geq 3$  and  $d$  such that there is a solution  $x \in [1/2 - 1/2^k, 1/2]$  to the equation (1.1). We use the following *one-step-replica-symmetry-breaking bound* proven in [58, Theorem E.3] for random regular graphs (see also [44]), which is the analog of [52, Theorem 3] for Erdős-Rényi graphs.

► **Theorem 2.1** (Theorem E.3 in [58]). *Let  $\mathcal{X}$  and  $S$  be finite sets and consider the partition function  $Z(G)$  (cf. Eq. (1.8)), where  $\theta$  in (1.6) satisfies the condition (1.7) for some  $\varepsilon > 0$  and  $b_s \geq 0$  holds for  $s \in S$ . Let  $\mathcal{M}_0 \equiv \mathcal{P}(\mathcal{X})$  be the space of probability measures over  $\mathcal{X}$ ,  $\mathcal{M}_1 \equiv \mathcal{P}(\mathcal{M}_0)$  be the space of probability measures over  $\mathcal{M}_0$ , and  $\mathcal{M}_2 \equiv \mathcal{P}(\mathcal{M}_1)$  be the space of probability measures over  $\mathcal{M}_1$ . For  $\zeta \in \mathcal{M}_2$ , let  $\underline{\eta} = (\eta_{a,j})_{a \geq 0, j \geq 0}$  be an array of i.i.d. samples from  $\zeta$ . For each index  $(a, j)$  let  $\rho_{a,j} \in \mathcal{P}(\mathcal{X})$  be a conditionally independent sample from  $\eta_{a,j}$ , and denote  $\underline{\rho} = (\rho_{a,j})_{a \geq 0, j \geq 0}$ . For  $x \in \mathcal{X}$  define random variables*

$$u_a(x) \equiv \sum_{\underline{x} \in \mathcal{X}^k} \mathbb{1}\{x_1 = x\} (1 - \theta_a(\underline{x})) \prod_{j=2}^k \rho_{a,j}(x_j), \quad u_a \equiv \sum_{\underline{x} \in \mathcal{X}^k} (1 - \theta_a(\underline{x})) \prod_{j=1}^k \rho_{a,j}(x_j),$$

where we recall that  $(\theta_a)_{a \geq 0}$  are i.i.d. copies of the random function  $\theta$ . For any  $\lambda \in (0, 1)$  and any  $\zeta \in \mathcal{M}_2$ ,

$F_n \leq \mathcal{P}(\zeta, \lambda) + O_\varepsilon(n^{-1/3})$ , where

$$\mathcal{P}(\zeta, \lambda) \equiv \mathcal{P}_\theta(\zeta, \lambda) := \lambda^{-1} \mathbb{E} \log \mathbb{E}' \left[ \left( \sum_{x \in \mathcal{X}} \prod_{a=1}^d u_a(x) \right)^\lambda \right] - (k-1) \alpha \lambda^{-1} \mathbb{E} \log \mathbb{E}' \left[ (u_0)^\lambda \right]. \quad (2.1)$$

Here,  $F_n$  is the free energy for the configuration model defined in (1.9),  $\mathbb{E}'$  denotes the expectation over  $\underline{\rho}$  conditioned on all else, and  $\mathbb{E}$  denotes the overall expectation.

► **Remark 2.1.** [58, Theorem E.3] is stated more general than Theorem 2.1 by considering independent *external field*  $\{h_v\}_{v \in V}$  and random  $(b_s)_{s \in S}$ . For our purposes, it suffices to consider non-random  $b_s \geq 0$  and  $h_v \equiv 1$ .

We use Theorem 2.1 for the positive temperature models in Definition 1.4. Note that  $\theta_{\text{NAE}}(\cdot; \beta)$  and  $\theta_{\text{COL}}(\cdot; \beta)$  satisfies the condition (1.7) with  $\varepsilon = e^{-\beta}$ . Furthermore, in the bound (2.1), we take  $\lambda = \beta^{-1/2}$  and  $\zeta \equiv \zeta_{k,d,\beta} \in \mathcal{P}(\mathcal{P}(\mathcal{P}(\{0,1\})))$  given by a point mass at  $\eta_{k,d,\beta}$ :

$$\zeta_{k,d,\beta} \equiv \delta_{\eta_{k,d,\beta}}, \quad (2.2)$$

where  $\eta_{k,d,\beta} \in \mathcal{P}(\mathcal{P}(\{0,1\}))$  is defined as follows. Identify  $\mathcal{P}(\{0,1\})$  with  $[0,1]$  by the map

$$\rho \in \mathcal{P}(\{0,1\}) \leftrightarrow \rho(1) \in [0,1].$$

Thus, denoting  $\eta \equiv \eta_{k,d,\beta} \in \mathcal{P}([0,1])$ , define

$$\eta \left( \frac{e^\beta}{e^\beta + e^{-\beta}} \right) = \eta \left( \frac{e^{-\beta}}{e^\beta + e^{-\beta}} \right) = x, \quad \eta \left( \frac{1}{2} \right) = 1 - 2x, \quad (2.3)$$

where  $x_\star \equiv x_\star(k,d)$  is the BP fixed point, i.e. the solution to the equation (1.1). Such choice of  $\zeta_{k,d,\beta}$  is motivated from physics [43] and previous mathematical works [31, Section 3] and [32, Section 4].

Before proceeding further, we show that if  $\zeta$  is given as in (2.2), (2.3), then  $\mathcal{P}(\zeta, \lambda)$  does not depend on literals. More precisely, suppose that  $\zeta = \delta_{\eta_0}$ , where  $\eta_0 \in \mathcal{P}([0,1])$  is such that  $\eta_0(dx) = \eta_0(d(1-x))$ , i.e.  $\rho \stackrel{d}{=} 1 - \rho$  holds for  $\rho \sim \eta_0$ . For a fixed  $\underline{\mathbf{L}} = (\mathbf{L}_i)_{i \leq k} \in \{0,1\}^k$ , let

$$\theta_{\underline{\mathbf{L}}}(\underline{x}) = (1 - e^{-\beta}) \cdot \left( \prod_{i=1}^k (\mathbf{L}_i \oplus x_i) + \prod_{i=1}^k (\mathbf{L}_i \oplus x_i \oplus 1) \right).$$

With abuse of notation, for  $x \in \{0,1\}$  and independent samples  $\rho_{a,j} \in \mathcal{P}(\{0,1\})$  from  $\eta_0$ , let

$$u_{a,\underline{\mathbf{L}}}(x) \equiv \sum_{\underline{x} \in \{0,1\}^k} \mathbb{1}\{x_1 = x\} (1 - \theta_{\underline{\mathbf{L}}}(\underline{x})) \prod_{j=2}^k \rho_{a,j}(x_j), \quad u_{\underline{\mathbf{L}}} \equiv \sum_{\underline{x} \in \{0,1\}^k} (1 - \theta_{\underline{\mathbf{L}}}(\underline{x})) \prod_{j=1}^k \rho_{0,j}(x_j),$$

where we consider  $\underline{\mathbf{L}} \in \{0,1\}^k$  to be fixed. Then, for a given sequence of literals  $\underline{\mathbf{L}}_a \in \{0,1\}^k$  for  $0 \leq a \leq d$ , let

$$\mathcal{P}(\delta_{\eta_0}, \lambda; (\underline{\mathbf{L}}_a)_{0 \leq a \leq d}) := \lambda^{-1} \log \mathbb{E}' \left( \sum_{x \in \{0,1\}} \prod_{a=1}^d u_{a,\underline{\mathbf{L}}_a}(x) \right)^\lambda - (k-1) \alpha \lambda^{-1} \mathbb{E} \log \mathbb{E}' (u_{\underline{\mathbf{L}}_0})^\lambda, \quad (2.4)$$

where  $\mathbb{E}'$  is the expectation with respect to the independent samples  $\rho_{a,j} \in \mathcal{P}(\{0,1\})$  from  $\eta_0$ . Note that if  $\underline{\mathbf{L}}_a \stackrel{i.i.d.}{\sim} \text{Unif}(\{0,1\}^k)$ , then  $\mathcal{P}_{\theta_{\text{NAE}}}(\delta_{\eta_0}, \lambda) = \mathbb{E}_{\underline{\mathbf{L}}} \mathcal{P}(\delta_{\eta_0}, \lambda; (\underline{\mathbf{L}}_a)_{0 \leq a \leq d})$  holds, and if  $\underline{\mathbf{L}}_a \equiv \underline{\mathbf{0}}$  for  $0 \leq a \leq d$ , then  $\mathcal{P}_{\theta_{\text{COL}}}(\delta_{\eta_0}, \lambda) = \mathcal{P}(\delta_{\eta_0}, \lambda; \underline{\mathbf{0}})$  holds. The following lemma then clarifies the mechanism behind the identical satisfiability upper bound in Theorem 1.1.

## 47:10 Upper Bounds on the Colorability Threshold for Random Regular Graph

► **Lemma 2.2.** Consider  $\zeta = \delta_{\eta_0}$  for some  $\eta_0 \in \mathcal{P}([0, 1])$  such that  $\eta_0(dx) = \eta_0(d(1-x))$ . Then, for any literals  $\underline{L}_a \in \{0, 1\}^k$  for  $0 \leq a \leq d$ , the value  $\mathcal{P}(\delta_{\eta_0}, \lambda; (\underline{L}_a)_{0 \leq a \leq d})$  does not depend on  $(\underline{L}_a)_{0 \leq a \leq d}$ . Thus,  $\mathcal{P}_{\theta_{\text{NAE}}}(\delta_{\eta_0}, \lambda) = \mathcal{P}_{\theta_{\text{COL}}}(\delta_{\eta_0}, \lambda)$  holds.

**Proof.** For fixed  $\underline{L}_a \in \{0, 1\}^k$  for  $0 \leq a \leq d$ , note that the vectors  $(u_{a, \underline{L}_a}(0), u_{a, \underline{L}_a}(1))$  are independent for  $0 \leq a \leq d$ . Thus, it suffices to show that for given  $\underline{L}, \underline{L}' \in \{0, 1\}^k$  and  $1 \leq a \leq d$ ,

$$u_{\underline{L}} \stackrel{d}{=} u_{\underline{L}'} \quad \text{and} \quad (u_{a, \underline{L}}(0), u_{a, \underline{L}}(1)) \stackrel{d}{=} (u_{a, \underline{L}'}(0), u_{a, \underline{L}'}(1)). \quad (2.5)$$

To this end, let  $\underline{L}' = \underline{0}$  and we first prove that  $u_{\underline{L}} \stackrel{d}{=} u_{\underline{0}}$  holds. Since  $\theta_{\underline{L}}(x) = \theta_{\underline{0}}(x \oplus \underline{L})$ ,

$$u_{\underline{L}} \equiv \sum_{\underline{x} \in \{0, 1\}^k} (1 - \theta_{\underline{L}}(\underline{x})) \prod_{j=1}^k \rho_{0,j}(x_j) = \sum_{\underline{x} \in \{0, 1\}^k} (1 - \theta_{\underline{0}}(\underline{x})) \prod_{j=1}^k \rho_{0,j}(x_j \oplus \underline{L}_j).$$

Note that since  $(\rho_{0,j})_{1 \leq j \leq k}$  are i.i.d. samples from  $\eta_0$  and  $\eta_0(dx) = \eta_0(d(1-x))$  holds, the sequence  $(\rho_{0,j}(\cdot \oplus \underline{L}_j))_{1 \leq j \leq k}$  are also i.i.d. from  $\eta_0$ . Hence, the equation above shows that  $u_{\underline{L}} \stackrel{d}{=} u_{\underline{0}}$  holds.

Next, we prove that  $(u_{a, \underline{L}}(0), u_{a, \underline{L}}(1)) \stackrel{d}{=} (u_{a, \underline{0}}(0), u_{a, \underline{0}}(1))$  holds. Without loss of generality, let  $a = 1$ . Again since  $\theta_{\underline{L}}(x) = \theta_{\underline{0}}(x \oplus \underline{L})$ ,

$$u_{1, \underline{L}}(x) = \sum_{\underline{x} \in \{0, 1\}^k} \mathbb{1}\{x_1 \oplus \underline{L}_1 = x\} (1 - \theta_{\underline{0}}(\underline{x})) \prod_{j=2}^k \rho_{1,j}(x_j \oplus \underline{L}_j)$$

Now, observe that  $\theta_{\underline{0}}(\cdot)$  is invariant under global flip, i.e.  $\theta_{\underline{0}}(x) = \theta_{\underline{0}}(x \oplus 1)$ . Thus, it follows that

$$u_{1, \underline{L}}(x) = \sum_{\underline{x} \in \{0, 1\}^k} \mathbb{1}\{x_1 = x\} [1 - \theta_{\underline{0}}(\underline{x})] \prod_{j=2}^k \rho_{1,j}(x_j \oplus \underline{L}_1 \oplus \underline{L}_j).$$

By the same reasons as above,  $(\rho_{1,j}(\cdot \oplus \underline{L}_1 \oplus \underline{L}_j))_{2 \leq j \leq k}$  have the same distribution as  $(\rho_{1,j})_{2 \leq j \leq k}$ , which are i.i.d. from  $\eta_0$ . Thus, we have that  $(u_{1, \underline{L}}(0), u_{1, \underline{L}}(1)) \stackrel{d}{=} (u_{1, \underline{0}}(0), u_{1, \underline{0}}(1))$ . Therefore, (2.5) holds, which concludes the proof. ◀

The following lemma relates  $\mathcal{P}_{\theta_{\text{COL}}}(\zeta_{k,d,\beta}, \beta^{-1/2}) = \mathcal{P}_{\theta_{\text{NAE}}}(\zeta_{k,d,\beta}, \beta^{-1/2})$ , and  $\star \Phi(d)$ , which plays a crucial role in proving Proposition 1.5. Recall the definition of  $\zeta_{k,d,\beta}$  in (2.2) and (2.3).

► **Lemma 2.3.**  $\mathcal{P}_{\theta_{\text{COL}}}(\zeta_{k,d,\beta}, \beta^{-1/2}) \leq C + \beta^{1/2} \times \star \Phi(d)$  holds for some constant  $C \in \mathbb{R}$ , which does not depend on  $\beta > 0$ .

**Proof.** Throughout, let  $(\rho_{a,j})_{a \geq 0, j \geq 0}$  denote i.i.d. samples from  $\eta_{k,d,\beta}$  defined in (2.3), and let  $\mathbb{E}'$  (resp.  $\mathbb{P}'$ ) denote the expectation (resp. probability) with respect to  $(\rho_{a,j})_{a \geq 0, j \geq 0}$ . Also, we use the generic notation  $C$  by a constant that does not depend on  $\beta > 0$ . Note that since  $\theta_{\text{COL}}$  and  $\eta_{k,d,\beta}$  are non-random, the outer expectation  $\mathbb{E}$  in the definition of  $\mathcal{P}(\zeta, \lambda)$  in (2.1) is redundant.

First, we bound the second term of the definition of  $\mathcal{P}_{\theta_{\text{COL}}}(\zeta_{k,d,\beta}, \beta^{-1/2})$  in (2.1):

$$\begin{aligned} & (k-1)\alpha\beta^{1/2} \log \mathbb{E}' \left[ (u_0)^{\beta^{-1/2}} \right] \\ &= (k-1)\alpha\beta^{1/2} \log \mathbb{E}' \left[ \left( 1 - (1 - e^{-\beta}) \left( \prod_{j=1}^k \rho_{0,j}(0) + \prod_{j=1}^k \rho_{0,j}(1) \right) \right)^{\beta^{-1/2}} \right] \end{aligned}$$

Note that the expectation inside the log in the right hand side above is bounded below by

$$2^{-\beta^{-1/2}} \cdot \mathbb{P}' \left( 1 - (1 - e^{-\beta}) \left( \prod_{j=1}^k \rho_{0,j}(0) + \prod_{j=1}^k \rho_{0,j}(1) \right) \geq \frac{1}{2} \right) = 2^{-\beta^{-1/2}} (1 - 2x^k),$$

where  $x$  is the solution to the BP equation (1.1) and the equality holds for large enough  $\beta \geq \beta_0$  since for large  $\beta$  and  $k \geq 3$ ,  $(1 - e^{-\beta}) \left( \prod_{j=1}^k \rho_{0,j}(0) + \prod_{j=1}^k \rho_{0,j}(1) \right) \geq \frac{1}{2}$  holds if and only if either  $\rho_{0,j}(1) = \frac{e^\beta}{e^\beta + e^{-\beta}}$  holds for all  $1 \leq j \leq k$ , or  $\rho_{0,j}(1) = \frac{e^{-\beta}}{e^\beta + e^{-\beta}}$  holds for all  $1 \leq j \leq k$ . Thus, it follows that

$$-(k-1)\alpha\lambda^{-1}\mathbb{E} \log \mathbb{E}' \left[ (u_0)^\lambda \right] \leq C - \beta^{1/2}(k-1)\alpha \log(1 - 2x^k). \tag{2.6}$$

Next, we estimate the first term of the definition of  $\mathcal{P}_{\theta_{\text{col}}}(\zeta_{k,d,\beta}, \beta^{-1/2})$  in (2.1), which equals

$$\begin{aligned} & \beta^{1/2} \log \mathbb{E}' \left[ \left( \sum_{x \in \{0,1\}} \prod_{a=1}^d u_a(x) \right)^{\beta^{-1/2}} \right] \\ &= \beta^{1/2} \log \mathbb{E}' \left[ \left( \prod_{a=1}^d \left( 1 - (1 - e^{-\beta}) \prod_{j=2}^k \rho_{a,j}(0) \right) + \prod_{a=1}^d \left( 1 - (1 - e^{-\beta}) \prod_{j=2}^k \rho_{a,j}(1) \right) \right)^{\beta^{-1/2}} \right] \end{aligned} \tag{2.7}$$

We upper bound the expectation inside the log in the above expression by

$$2^{\beta^{-1/2}} \cdot \mathbb{P}'(\mathcal{A}) + (3e^{-\beta})^{\beta^{-1/2}},$$

where

$$\mathcal{A} := \left\{ \prod_{a=1}^d \left( 1 - (1 - e^{-\beta}) \prod_{j=2}^k \rho_{a,j}(0) \right) + \prod_{a=1}^d \left( 1 - (1 - e^{-\beta}) \prod_{j=2}^k \rho_{a,j}(1) \right) \geq 3e^{-\beta} \right\}.$$

Define the events  $\mathcal{E}_0$  and  $\mathcal{E}_1$  involving  $(\rho_{a,j})_{1 \leq a \leq d, 2 \leq j \leq k}$  as follows.

- $\mathcal{E}_0$  is the event such that for each  $1 \leq a \leq d$ , we have for some  $j \in \{2, \dots, k\}$  that  $\rho_{a,j}(0) \neq \frac{e^\beta}{e^\beta + e^{-\beta}}$ .
- $\mathcal{E}_1$  is the event such that for each  $1 \leq a \leq d$ , we have for some  $j \in \{2, \dots, k\}$  that  $\rho_{a,j}(1) \neq \frac{e^\beta}{e^\beta + e^{-\beta}}$ .

We now claim that for large enough  $\beta$ , the event  $\mathcal{A}$  is included in  $\mathcal{E}_0 \cup \mathcal{E}_1$ . To this end, suppose that the event  $(\mathcal{E}_0 \cup \mathcal{E}_1)^c = \mathcal{E}_0^c \cap \mathcal{E}_1^c$  holds. Then, for each  $x \in \{0, 1\}$ , for some  $a \equiv a(x) \in \{1, \dots, d\}$  such that  $\rho_{a,j}(x) = \frac{e^\beta}{e^\beta + e^{-\beta}}$  holds for all  $2 \leq j \leq k$ . Thus, for  $x \in \{0, 1\}$ , we have

$$\prod_{a=1}^d \left( 1 - (1 - e^{-\beta}) \prod_{j=2}^k \rho_{a,j}(x) \right) \leq 1 - (1 - e^{-\beta}) \left( \frac{e^\beta}{e^\beta + e^{-\beta}} \right)^{k-1} \leq e^{-\beta} + ke^{-2\beta} < \frac{3}{2}e^{-\beta},$$

where the last inequality holds for large enough  $\beta \geq \beta_k$  and we used  $(1-x)^{k-1} \geq 1 - (k-1)x$  for  $x > 0$  in the second inequality. Hence, summing over  $x \in \{0, 1\}$  gives that the event  $\mathcal{A}$  cannot hold, which proves our claim that  $\mathcal{A} \subset \mathcal{E}_0 \cup \mathcal{E}_1$ . Consequently, the term (2.7) is bounded above by

$$\beta^{1/2} \log \left( 2^{\beta^{-1/2}} \cdot \mathbb{P}'(\mathcal{E}_0 \cup \mathcal{E}_1) + (3e^{-\beta})^{\beta^{-1/2}} \right) \leq \beta^{1/2} \log \mathbb{P}'(\mathcal{E}_0 \cup \mathcal{E}_1) + C.$$

## 47:12 Upper Bounds on the Colorability Threshold for Random Regular Graph

Note that  $\mathbb{P}'(\mathcal{E}_0 \cup \mathcal{E}_1)$  can be calculated explicitly by

$$\mathbb{P}'(\mathcal{E}_0 \cup \mathcal{E}_1) = 2(1 - x^{k-1})^d - (1 - 2x^{k-1})^d = \frac{(1 - x^{k-1})^{d-1}(1 - 2x^k)}{1 - x},$$

where in the final equality, we used the fact that  $x$  is the solution to the equation (1.1). Therefore, we have proven that

$$\begin{aligned} \beta^{1/2} \log \mathbb{E}' \left[ \left( \sum_{x \in \{0,1\}} \prod_{a=1}^d u_a(x) \right)^{\beta^{-1/2}} \right] \\ \leq C + \beta^{1/2} (-\log(1-x) + (d-1) \log(1-x^{k-1}) + \log(1-2x^k)). \end{aligned} \quad (2.8)$$

In conclusion, combining (2.6) and (2.8), and recalling the definition of  $\star\Phi(d)$  in (1.2), we have

$$\mathcal{P}_{\theta_{\text{COL}}}(\zeta_{k,d,\beta}, \beta^{-1/2}) \leq C + \beta^{1/2} \star\Phi(d),$$

which concludes the proof.  $\blacktriangleleft$

**Proof of Proposition 1.5.** Given a NAE-SAT instance  $\mathcal{G}$ , let  $\text{SOL}(\mathcal{G}) \subset \{0,1\}^V$  denotes the set of NAE-SAT solutions. Also, let  $Z_{\beta, \text{NAE}}(\mathcal{G})$  denotes the partition function (1.8) for  $\theta = \theta_{\text{NAE}}(\cdot; \beta)$ . Note that if  $\underline{x} \in \text{SOL}(\mathcal{G})$ , then  $\theta_{\text{NAE}}(\underline{x}_{\delta_a}) = 0$  for any  $a \in F$ , thus we have for any  $\beta > 0$  that

$$Z_{\beta, \text{NAE}}(\mathcal{G}) \equiv \sum_{\underline{x} \in \{0,1\}^V} \prod_{a \in F} (1 - \theta_{\text{NAE}}(\underline{x}_{\delta_a}; \beta)) \geq |\text{SOL}(\mathcal{G})|. \quad (2.9)$$

On the other hand, since  $\theta_{\text{NAE}}(\cdot; \beta)$  satisfies the condition (1.7) with  $\varepsilon = e^{-\beta}$ , we have by Theorem 2.1 that

$$\frac{1}{n} \mathbb{E} \left[ \log Z_{\beta, \text{NAE}}(\mathcal{G}) \right] \leq \mathcal{P}_{\theta_{\text{NAE}}}(\zeta_{k,d,\beta}, \beta^{-1/2}) + o_n(1) = \mathcal{P}_{\theta_{\text{COL}}}(\zeta_{k,d,\beta}, \beta^{-1/2}) + o_n(1),$$

where the last equality is due to Lemma 2.2. By Lemma 2.3, the right hand side is further bounded by

$$\frac{1}{n} \mathbb{E} \left[ \log Z_{\beta, \text{NAE}}(\mathcal{G}) \right] \leq \beta^{1/2} \cdot \star\Phi(d) + C + o_n(1),$$

for some constant  $C$  that does not depend on  $n$  nor  $\beta$ . If  $\star\Phi(d) < 0$ , then for large enough  $\beta > 0$ ,  $\beta^{1/2} \cdot \star\Phi(d) + C < -1$  holds, thus  $n^{-1} \mathbb{E} \left[ \log Z_{\beta, \text{NAE}}(\mathcal{G}) \right] < -1$  holds for large enough  $n$ . For such  $\beta = \beta_0(k, d) > 0$ , we have by (2.9) and Lemma 1.3 that for large enough  $n$ ,

$$\mathbb{P} \left( |\text{SOL}(\mathcal{G})| \geq 1 \right) \leq \mathbb{P} \left( \left| \frac{1}{n} \log Z_{\beta_0, \text{NAE}}(\mathcal{G}) - \frac{1}{n} \mathbb{E} \left[ \log Z_{\beta_0, \text{NAE}}(\mathcal{G}) \right] \right| \geq 1 \right) \leq e^{-cn},$$

for some constant  $c$  that depends only on  $\beta_0 > 0$ , which finishes the proof for the NAE-SAT model.

Given a configuration model  $\mathbf{G}$ , let  $Z_{\beta, \text{COL}}(\mathbf{G})$  denote the partition function (1.8) for  $\theta = \theta_{\text{COL}}(\cdot; \beta)$ . Then, by the same reasoning, Theorem 2.1 and Lemma 2.3 shows that if  $\star\Phi(d) < 0$  then  $\frac{1}{n} \mathbb{E} \left[ \log Z_{\beta, \text{COL}}(\mathbf{G}) \right] < -1$  holds for large enough  $\beta = \beta_0(k, d) > 0$  and  $n$  large enough. On the event that there exists a 2-coloring on  $\mathbf{G}$ ,  $Z_{\beta, \text{COL}}(\mathbf{G}) \geq 1$  holds, so Lemma 1.3 again concludes the proof.  $\blacktriangleleft$

## 2.2 Proof of Lemma 1.3

Recall that  $\mathbf{G} = (V, F, \mathbf{E})$  is generated from the configuration model, where the  $\mathbf{E}$  is drawn uniformly from  $S_{nd}$ . Thus,  $\mathbf{E}$  has the same law as sequentially drawing random clauses  $\mathbf{a}_1, \dots, \mathbf{a}_m$  as follows. At times  $t \in \{1, \dots, m\}$  clause  $\mathbf{a}_t$  is drawn by connecting the  $k$  adjacent half-edges to previously unmatched half-edges adjacent to variables. For  $1 \leq t \leq m$ , let  $\mathcal{F}_t$  be the  $\sigma$ -algebra generated by  $\mathbf{a}_1, \dots, \mathbf{a}_t$ , and  $\mathcal{F}_0 \equiv \emptyset$ . Denote  $M_t \equiv \mathbb{E}[\log Z(\mathbf{G}) \mid \mathcal{F}_t]$  by the associated Doob martingale. Note that if  $\mathbf{G} = (V, F, E)$  and  $\mathbf{G}' = (V, F, E')$  has the same set of edges except for those adjacent to two clauses  $a_1 \neq a_2 \in F$ , then by our assumption of  $\theta$  in (1.7) and the definition of  $Z(G)$  in (1.8), it follows that  $\varepsilon^2 \leq Z(G)/Z(G') \leq \varepsilon^{-2}$  holds. Thus, we have for every  $t \in \{0, 1, \dots, m-1\}$  that

$$\left| M_{t+1} - M_t \right| \equiv \left| \mathbb{E}[\log Z(\mathbf{G}) \mid \mathcal{F}_{t+1}] - \mathbb{E}[\log Z(\mathbf{G}) \mid \mathcal{F}_t] \right| \leq 2 \log(1/\varepsilon), \quad (2.10)$$

from which Lemma 1.3 follows.

**Proof of Lemma 1.3.** Note that  $M_m = \log Z(\mathbf{G})$  and  $M_0 = \mathbb{E}[\log Z(\mathbf{G})]$  holds and  $(M_t)_{0 \leq t \leq m}$  is a martingale with bounded difference by (2.10). Therefore, the conclusion follows from Azuma Hoeffding's inequality.  $\blacktriangleleft$

## 2.3 Proof of Lemma 1.7

The following notations are convenient for the proof of Lemma 1.7. For non-negative quantities  $f = f_{d,k,n}$  and  $g = g_{d,k,n}$ , we use any of the equivalent notations  $f = O_{k,d}(g)$ ,  $g = \Omega_{k,d}(f)$ ,  $f \lesssim_{k,d} g$  and  $g \gtrsim_{k,d} f$  to indicate that there exists a constant  $C_{k,d}$ , which only depends on  $k, d$  such that  $f \leq C_{k,d} \cdot g$ . We drop the subscripts  $d$  (resp.  $k, d$ ) if the constant  $C_{k,d}$  does not depend on  $d$  (resp.  $k, d$ ). When  $f \lesssim_{k,d} g$  and  $g \lesssim_{k,d} f$ , we write  $f \asymp_{k,d} g$ . Similarly when  $f \lesssim g$  and  $g \lesssim f$ , we write  $f \asymp g$ .

Note that  $\mathbb{E}Z_{\text{COL}}$  is the sum over  $\underline{x} \in \{0, 1\}^V$  of the probabilities that  $\underline{x}$  is a 2-coloring on  $\mathbf{G}$ . By symmetry, the probability of  $\underline{x} \in \{0, 1\}^V$  being a 2-coloring depends only on the number  $n\gamma$  of nodes having color 1, which we denote by  $\mathbf{p}_\gamma$ . Thus,  $\mathbb{E}Z_{\text{COL}} = \sum_\gamma \binom{n}{n\gamma} \mathbf{p}_\gamma$ , where the sum is over  $\gamma \in (0, 1)$  such that  $n\gamma \in \mathbb{Z}$ . Moreover, we can express  $\mathbf{p}_\gamma$  as follows. Let  $X_1, \dots, X_m$  be i.i.d. Binom( $k, \gamma$ ) random variables and denote  $\mathbb{P}_\gamma$  by the probability with respect to  $(X_i)_{i \leq m}$ . Then, we have

$$\begin{aligned} \mathbf{p}_\gamma &= \mathbb{P}_\gamma \left( X_i \notin \{0, k\} \text{ for all } 1 \leq i \leq m \mid \sum_{i=1}^m X_i = km\gamma \right) \\ &\leq \frac{\mathbb{P}_\gamma(X_i \notin \{0, k\} \text{ for all } 1 \leq i \leq m)}{\mathbb{P}_\gamma(\sum_{i=1}^m X_i = km\gamma)} \lesssim_k \sqrt{m} (1 - \gamma^k - (1 - \gamma)^k)^m, \end{aligned} \quad (2.11)$$

where the last inequality is due to a Stirling's approximation. It follows that

$$\begin{aligned} \mathbb{E}Z_{\text{COL}} &\leq n^{O(1)} \sum_\gamma \exp(nF_\alpha(\gamma)), \quad \text{where} \\ F_\alpha(\gamma) &:= H(\gamma) + \alpha \log(1 - \gamma^k - (1 - \gamma)^k). \end{aligned} \quad (2.12)$$

Here,  $H(\gamma) \equiv -\gamma \log \gamma - (1 - \gamma) \log(1 - \gamma)$  is the entropy of  $\gamma$ . Note that  $\gamma \rightarrow \gamma^k + (1 - \gamma)^k$  is uniquely minimized at  $\gamma = 1/2$ . Further, the entropy  $H(\gamma)$  is strictly concave and is maximized at  $\gamma = 1/2$ . Thus,  $\gamma \rightarrow F_\alpha(\gamma)$  is uniquely maximized at  $\gamma = 1/2$  with  $\frac{\partial^2 F_\alpha}{\partial \gamma^2}(1/2) < 0$ . Since  $\mathbb{E}Z_{\text{NAE}} = \exp(nF_\alpha(1/2))$ , it follows from (2.12) that

$$\mathbb{E}Z_{\text{COL}} \leq n^{O(1)} \exp(nF_\alpha(1/2)) = n^{O(1)} \cdot \mathbb{E}Z_{\text{NAE}}. \quad (2.13)$$

We now show that the polynomial factor  $n^{O(1)}$  can actually be removed with a matching lower bound.

## 47:14 Upper Bounds on the Colorability Threshold for Random Regular Graph

First, by (2.11) and the fact that  $\gamma \rightarrow F_\alpha(\gamma)$  is uniquely maximized at  $\gamma = 1/2$  with strictly negative second derivative, the contribution to  $\mathbb{E}Z_{\text{COL}}$  from  $\gamma$  such that  $|\gamma - 1/2| \geq n^{-1/3}$  is negligible:

$$\sum_{|\gamma - 1/2| \geq n^{-1/3}} \binom{n}{n\gamma} \mathbf{p}_\gamma \lesssim_{k,d} \exp(-\Omega_{k,d}(n^{1/3})) \cdot \mathbb{E}Z_{\text{NAE}}. \quad (2.14)$$

Thus, we focus on the regime  $|\gamma - 1/2| \leq n^{-1/3}$ . Note that we can calculate  $\mathbf{p}_\gamma$  by summing over the empirical distribution  $\nu$  of  $(X_i)_{i \leq m}$ . Consider  $\nu \in \mathcal{P}(\{1, \dots, k-1\})$  and let  $p_\gamma(j) := \binom{k}{j} \gamma^j (1-\gamma)^{k-j}$ . Then,

$$\mathbf{p}_\gamma = \frac{\sum_\nu \mathbb{1}\left(\sum_j j\nu_j = km\gamma\right) e^{-km\gamma\lambda} \binom{m}{m\nu} \prod_j (p_\gamma(j)e^{\lambda j})^{m\nu_j}}{\mathbb{P}_\gamma\left(\sum_{i=1}^m X_i = km\gamma\right)},$$

where  $\binom{m}{m\nu} \equiv \frac{m!}{\prod_j (m\nu_j)!}$  and we introduced a lagrange parameter  $\lambda \in \mathbb{R}$  in the last equality. Let

$$\nu_{\gamma,\lambda}(x) := \frac{p_\gamma(x)e^{\lambda x}}{\sum_{j=1}^{k-1} p_\gamma(j)e^{\lambda j}} \quad \text{for } 1 \leq x \leq k-1,$$

and denote  $\mathbb{P}_{\gamma,\lambda}$  by the probability with respect to  $\tilde{X}_1, \dots, \tilde{X}_m \stackrel{i.i.d.}{\sim} \nu_{\gamma,\lambda}$ . Then, it follows that

$$\mathbf{p}_\gamma = \frac{\mathbb{P}_{\gamma,\lambda}\left(\sum_{i=1}^m \tilde{X}_i = km\gamma\right)}{\mathbb{P}_\gamma\left(\sum_{i=1}^m X_i = km\gamma\right)} \exp(-m \cdot \Xi(\gamma, \lambda)), \quad \text{where } \Xi(\gamma, \lambda) := k\gamma\lambda - \log\left(\sum_{j=1}^{k-1} p_\gamma(j)e^{\lambda j}\right). \quad (2.15)$$

In order to use the local central limit theorem, we take  $\lambda = \lambda(\gamma)$  such that  $\mathbb{E}_{\gamma,\lambda}\tilde{X} = k\gamma$ , where  $\tilde{X} \sim \nu_{\gamma,\lambda}$ . The existence of such  $\lambda(\gamma)$  is guaranteed by the lemma below.

► **Lemma 2.4.** *For large enough  $n$  and all  $\gamma$  such that  $|\gamma - 1/2| \leq n^{-1/3}$ , there exists a unique  $\lambda = \lambda(\gamma)$  such that  $\mathbb{E}_{\gamma,\lambda}\tilde{X} = k\gamma$  holds. Furthermore, we have  $\lambda(1/2) = 0$  and  $|\lambda(\gamma)| \lesssim_k n^{-1/3}$  holds uniformly over  $|\gamma - 1/2| \leq n^{-1/3}$ .*

**Proof.** Note that we have  $\frac{\partial \Xi}{\partial \lambda}(\gamma, \lambda) = k\gamma - \mathbb{E}_{\gamma,\lambda}\tilde{X}$  by definition of  $\nu_{\gamma,\lambda}$  and  $\Xi(\gamma, \lambda)$ . Further, we have that

$$\frac{\partial \Xi}{\partial \lambda}\left(\frac{1}{2}, 0\right) = \frac{k}{2} - \mathbb{E}_{\frac{1}{2}, 0}\tilde{X} = \frac{k}{2} - \mathbb{E}_{\frac{1}{2}}[X | X \notin \{0, k\}] = 0,$$

where  $\mathbb{E}_{\frac{1}{2}}$  is with respect to  $X \sim \text{Binom}(1/2)$ . Since  $\lambda \rightarrow \log\left(\sum_{j=1}^{k-1} p_\gamma(j)e^{\lambda j}\right)$  is strongly convex, we have  $\frac{\partial^2 \Xi}{\partial \lambda^2}\left(\frac{1}{2}, 0\right) < 0$ . Thus, implicit function theorem shows that for  $\gamma \in (1/2 - \varepsilon, 1/2 + \varepsilon)$ , where  $\varepsilon = \varepsilon(k) > 0$  depends only on  $k$ , there exists  $\lambda = \lambda(\gamma)$  such that  $\frac{\partial \Xi}{\partial \lambda}(\gamma, \lambda(\gamma)) = 0$  holds, and that  $\gamma \rightarrow \lambda(\gamma)$  is continuously differentiable. Therefore, for large enough  $n$  and  $\gamma \in (1/2 - n^{-1/3}, 1/2 + n^{-1/3})$ , there exists a unique  $\lambda = \lambda(\gamma)$  such that  $\mathbb{E}_{\gamma,\lambda(\gamma)}\tilde{X} = k\gamma$ , and  $|\lambda(\gamma)| \lesssim_k n^{-1/3}$  holds uniformly over  $\gamma \in (1/2 - n^{-1/3}, 1/2 + n^{-1/3})$ . ◀

Having Lemma 2.4 in hand, we prove Lemma 1.7 by appealing to the local central limit theorem.



**Proof of Lemma 1.7.** The contribution to  $\mathbb{E}Z_{\text{COL}}$  from  $\gamma$  such that  $|\gamma - 1/2| \geq n^{-1/3}$  is negligible by (2.14), thus we consider  $\gamma$  such that  $|\gamma - 1/2| \leq n^{-1/3}$  holds. To this end, we take  $\lambda = \lambda(\gamma)$  from Lemma 2.4 in equation (2.15). Then, by the local central limit theorem [13],

$$\mathbf{p}_\gamma \asymp \left( \frac{\text{Var}_\gamma(X)}{\text{Var}_{\gamma, \lambda(\gamma)}(\tilde{X})} \right)^{1/2} \cdot \exp\left(-m \cdot \Xi(\gamma, \lambda(\gamma))\right), \quad (2.16)$$

where  $X \sim \text{Binom}(k, \gamma)$  and  $\tilde{X} \sim \nu_{\gamma, \lambda(\gamma)}$ . Lemma 2.4 further shows that  $|\lambda(\gamma)| \lesssim_k n^{-1/3}$ , thus we have

$$\text{Var}_{\gamma, \lambda(\gamma)}(\tilde{X}) \asymp_k \text{Var}_\gamma(X \mid 1 \leq X \leq k-1) \asymp_k \text{Var}_\gamma(X), \quad (2.17)$$

where the final estimate holds because  $|\gamma - 1/2| \leq n^{-1/3}$ . Combining with (2.14), it follows that

$$\mathbb{E}Z_{\text{COL}} = (1 + o_n(1)) \sum_{|\gamma - 1/2| \leq n^{-1/3}} \binom{n}{n\gamma} \mathbf{p}_\gamma \asymp_{k,d} n^{-1/2} \sum_{|\gamma - 1/2| \leq n^{-1/3}} \exp(nG_\alpha(\gamma)), \quad (2.18)$$

where

$$G_\alpha(\gamma) := H(\gamma) - \alpha \cdot \Xi(\gamma, \lambda(\gamma)).$$

Note that by comparing (2.16) and (2.17) with (2.11), we have  $G_\alpha(\gamma) \leq F_\alpha(\gamma)$  for  $|\gamma - 1/2| \leq n^{-1/3}$ . Also, note that for  $\gamma = 1/2$ ,  $G_\alpha(1/2) = F_\alpha(1/2)$  holds since

$$G_\alpha(1/2) = H(1/2) - \alpha \cdot \Xi(1/2, 0) = H(1/2) + \alpha \log(1 - \gamma^k - (1 - \gamma)^k),$$

where we used  $\lambda(1/2) = 0$  by Lemma 2.4. Recalling that  $\gamma \rightarrow F_\alpha(\gamma)$  is uniquely maximized at  $\gamma = 1/2$  with strictly negative second derivative at the maximizer, it follows that the same holds for  $\gamma \rightarrow G_\alpha(\gamma)$ . Therefore, the sum in the right hand side of (2.18) can be approximated by a Gaussian integration, which shows that

$$\mathbb{E}Z_{\text{COL}} \asymp_{k,d} \exp(nG_\alpha(1/2)) = \mathbb{E}Z_{\text{NAE}}.$$

This concludes the proof. ◀

### 3 Proof of Proposition 1.6

In this section, we prove Proposition 1.6 for  $k \geq 4$ . The proof of Proposition 1.6 for  $k = 3$  is available in the arXiv version [14]. The proof of Proposition 1.6 for  $k \geq 4$  can be split into the following two lemmas. In Section 3.1, we prove Lemma 3.1 which guarantees the existence and the uniqueness of the BP fixed point for  $k \geq 4$ .

► **Lemma 3.1.** *For  $k \geq 4$  and  $d \in [d_{\text{lb}}(k), d_{\text{ub}}(k)]$ , there exists a unique solution to  $\Psi_d(x) = x$  in the range  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ .*

By Lemma 3.1, the function  $d \rightarrow \star\Phi(d)$  is well-defined. In Section 3.2, we prove Lemma 3.2 which guarantees that  $d_\star(k)$  is well-defined for  $k \geq 4$ .

► **Lemma 3.2.** *For  $k \geq 4$ , the function  $d \rightarrow \star\Phi(d)$  is continuous for  $d \in [d_{\text{lb}}(k), d_{\text{ub}}(k)]$ . Further,  $\star\Phi(d_{\text{lb}}(k)) > 0$  and  $\star\Phi(d_{\text{ub}}(k)) < 0$  hold.*

**Proof of Proposition 1.6 for  $k \geq 4$ .** This is immediate from Lemma 3.1 and Lemma 3.2. ◀

### 3.1 Proof of Lemma 3.1

Recall the variable BP recursion  $\dot{\Psi}$  and the clause BP recursion  $\hat{\Psi}$  defined in (1.12). To prove the uniqueness of the BP fixed point, we show that the BP recursion  $\Psi_d \equiv \dot{\Psi} \circ \hat{\Psi}$  is a contraction for  $k \geq 4$ .

► **Lemma 3.3.** *For  $k \geq 4$  and  $d \in [d_{\text{lb}}(k), d_{\text{ub}}(k)]$ ,  $|(\Psi_d)'(x)| < 1$  holds uniformly over  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ .*

**Proof.** Throughout, we let  $x \in [1/2 - 1/2^k, 1/2]$  and denote  $v = \hat{\Psi}(x)$ . We first consider  $k \geq 5$ . Observe that the derivative of the clause BP recursion can simply be bounded in absolute value by

$$|(\hat{\Psi})'(x)| = \frac{(k-1)x^{k-2}}{(1-x^{k-1})^2} \leq \frac{(k-1) \cdot 2^{-k+2}}{(1-2^{-k+1})^2} = \frac{4(k-1)}{2^k(1-2^{-k+1})^2}, \quad (3.1)$$

where the inequality holds since  $x \rightarrow \frac{x^{k-2}}{(1-x^{k-1})^2}$  is increasing. Similarly, we bound the derivative of the variable BP recursion:

$$|(\dot{\Psi})'(v)| = \frac{(d-1)v^{d-2}}{(2-v^{d-1})^2} \leq \frac{(d-1)v_0^{d-2}}{(2-v_0^{d-1})^2} \leq \frac{(d-1)v_0^{d-2}}{(2-v_0^{d-2})^2}, \quad (3.2)$$

where we denoted  $v_0 := \hat{\Psi}(x_0)$  for  $x_0 = 1/2 - 1/2^k$ . The first inequality holds because  $x \rightarrow \hat{\Psi}(\cdot)$  is decreasing on  $[1/2 - 1/2^k, 1/2]$ , and the last inequality holds since  $v_0 < 1$ . To this end, we upper bound  $v_0^{d-2}$  by

$$v_0^{d-2} = \left(1 - \frac{x_0^{k-1}}{1-x_0^{k-1}}\right)^{d-2} \leq (1-x_0^{k-1})^{d-2} \leq e^{-(d-2)x_0^{k-1}}. \quad (3.3)$$

Note that  $x_0^{k-1} = (\frac{1}{2})^{k-1} (1 - \frac{2}{2^k})^{k-1} \geq (\frac{1}{2})^{k-1} (1 - \frac{2(k-1)}{2^k})$  and  $d \geq (2^{k-1} - 2)k \log 2$  hold, thus we can lower bound  $(d-2)x_0^{k-1}$  by

$$(d-2)x_0^{k-1} \geq \left(k \log 2 - \frac{4k \log 2 + 4}{2^k}\right) \cdot \left(1 - \frac{2(k-1)}{2^k}\right).$$

Thus, combining with (3.3) shows that

$$v_0^{d-2} \leq 2^{-k} e^{\varepsilon_k}, \quad \text{where } \varepsilon_k := \frac{2(k-1)k \log 2}{2^k} + \frac{4k \log 2 + 4}{2^k} \left(1 - \frac{2(k-1)}{2^k}\right). \quad (3.4)$$

Plugging this bound into (3.2), we have

$$|(\dot{\Psi})'(v)| < (d-1) \frac{v_0^{d-2}}{(2-v_0^{d-2})^2} \leq (2^{k-1}k \log 2 - 1) \cdot \frac{2^{-k} \cdot e^{\varepsilon_k}}{(2-2^{-k}e^{\varepsilon_k})^2}.$$

Combining with the contraction of clause BP recursion in (3.1), we have

$$|(\Psi_d)'(x)| \leq \alpha_k := \frac{2k(k-1) \log 2}{2^k} \cdot \left(1 - \frac{1}{2^{k-1}k \log 2}\right) \cdot \frac{e^{\varepsilon_k}}{(1-2^{-k+1})^2(2-2^{-k}e^{\varepsilon_k})^2}.$$

By comparing  $\varepsilon_k$  and  $\varepsilon_{k+1}$  for  $k \geq 5$ , it can be easily checked that  $k \rightarrow \varepsilon_k$  is decreasing, and the same holds for  $k \rightarrow \frac{2k(k-1) \log 2}{2^k} \cdot \left(1 - \frac{1}{2^{k-1}k \log 2}\right)$ . Thus,  $k \rightarrow \alpha_k$  is decreasing for  $k \geq 5$ . Furthermore,  $\alpha_5$  can be calculated up to arbitrary precision (e.g. by Mathematica), which satisfies  $\alpha_5 < 0.99 < 1$ . Consequently,  $|(\Psi_d)'(x)| < 1$  holds for  $k \geq 5$ .

The case where  $k = 4$  is more delicate, and the previous strategy of bounding the derivative of clause and variable BP recursions separately no longer is successful. To this end, we bound  $(\Psi_d)'(x)$  directly. If we denote  $v = \hat{\Psi}_k(x)$ , then

$$|(\Psi_d)'(x)| = |(\hat{\Psi})'(x)| \cdot |(\dot{\Psi})'(v)| = \frac{(k-1)(d-1)v^{d-2}}{(2-v^{d-1})^2} \cdot \frac{x^{k-1}}{(1-x^{k-1})^2} \cdot \frac{1}{x}.$$

Since  $v \equiv \hat{\Psi}_k(x) \equiv \frac{1-2x^{k-1}}{1-x^{k-1}}$ , rearranging gives  $x^{k-1} = \frac{1-v}{2-v}$ . Substituting this in for  $x^{k-1}$ , we have that

$$|(\Psi_d)'(x)| = (k-1)(d-1) \cdot \frac{v^{d-2}(2-v)(1-v)}{(2-v^{d-1})^2} \cdot \frac{1}{x}. \quad (3.5)$$

We now claim that  $v \rightarrow \frac{v^{d-2}(2-v)(1-v)}{(2-v^{d-1})^2}$  is increasing for  $v \in [\hat{\Psi}_4(1/2), \hat{\Psi}_4(1/2 - 1/2^4)]$  and  $d \in [24 \log 2, 32 \log 2]$  (recall that  $24 \log 2 > 16.7 \equiv d_{\text{ibd}}(4)$  holds). Since  $v \rightarrow (2-v^{d-1})^2$  is decreasing, it suffices to show that  $v \rightarrow v^{d-2}(2-v)(1-v)$  is increasing. Note that

$$\frac{d}{dv} \left( v^{d-2}(2-v)(1-v) \right) = (dv^2 - 3(d-1)v + 2(d-2))v^{d-3} > 0 \iff d > \frac{4-3v}{(2-v)(1-v)}.$$

Note that  $v \rightarrow \frac{4-3v}{(2-v)(1-v)}$  is increasing since its derivative is given by  $\frac{3v^2-8v+6}{(2-v)^2(1-v)^2} > 0$ . Thus, to prove our claim, it suffices to check that for  $d_0 := 24 \log 2$  and  $v_0 = \hat{\Psi}_4(1/2 - 1/2^4)$  that  $d_0 > \frac{4-3v_0}{(2-v_0)(1-v_0)}$  holds. By a direct calculation,  $v_0 = 3410/3753 < 0.91$  and  $24 \log 2 > 16 > \frac{4-3 \cdot 0.91}{(2-0.91)(1-0.91)}$  holds, thus the claim that  $v \rightarrow \frac{v^{d-2}(2-v)(1-v)}{(2-v^{d-1})^2}$  is increasing is proven for  $d, v$  in the regime of interest.

Note that  $x \rightarrow v = \hat{\Psi}_4(x)$  is decreasing, thus (3.5) and our previous claim shows that for all  $x_0 \leq x \leq 1/2$ , where  $x_0 = 1/2 - 1/2^4$ , we have

$$|(\Psi_d)'(x)| \leq (d-1)(k-1) \frac{v_0^{d-2}(2-v_0)(1-v_0)}{(2-v_0^{d-1})^2} \cdot \frac{1}{x_0},$$

where  $v_0 = \hat{\Psi}_4(x_0) = 3410/3753$ . We next show that the right hand side as a function of  $d \in [24 \log 2, 32 \log 2]$  is decreasing: since  $d \rightarrow (2-v_0^{d-1})^2$  is increasing, it suffices to show that  $d \rightarrow (d-1)v_0^{d-2}$  is decreasing. Note that

$$\frac{d}{dd} \left( (d-1)v_0^{d-2} \right) = v_0^{d-2} \left( 1 - (d-1) \log(1/v_0) \right) < 0 \iff d > \frac{1}{\log(1/v_0)} + 1,$$

and it can be verified that  $24 \log 2 > 16 > 1/\log(3753/3410) + 1$  holds. Therefore, for  $k = 4$ , it follows that for  $d_0 = 24 \log 2$ ,

$$|(\Psi_d)'(x)| \leq 3(d_0-1) \frac{v_0^{d_0-2}(2-v_0)(1-v_0)}{(2-v_0^{d_0-1})^2} \cdot \frac{1}{x_0}.$$

The right hand side can be computed to arbitrary precision (e.g. by Mathematica), it can be verified that  $3(d_0-1) \frac{v_0^{d_0-2}(2-v_0)(1-v_0)}{(2-v_0^{d_0-1})^2} \cdot \frac{1}{x_0} < 0.9 < 1$ . This concludes the proof for the case  $k = 4$ .  $\blacktriangleleft$

In the proof of Lemma 3.3, we did not use the adjustment for  $d_{\text{ibd}}(4) \equiv 16.7 > 24 \log 2$ . That is,  $\max_{\frac{1}{2} - \frac{1}{2^4} \leq x \leq \frac{1}{2}} |(\Psi_d)'(x)| < 1$  holds for  $d \in [24 \log 2, 32 \log 2]$ . The adjustment  $d_{\text{ibd}}(4) \equiv 16.7$  is needed for the following lemma, which guarantees the existence of the solution to  $\Psi_d(x) = x$ .

► **Lemma 3.4.**  $\Psi_d(\frac{1}{2} - \frac{1}{2^k}) > \frac{1}{2} - \frac{1}{2^k}$  holds for  $k \geq 4$  for  $d \in [d_{\text{lb}d}(k), d_{\text{ub}d}(k)]$ .

**Proof.** Let  $v_0 \equiv v_0(k) = \hat{\Psi}(\frac{1}{2} - \frac{1}{2^k})$  as before. Then, from the definition of  $\hat{\Psi}, \hat{\Psi}$  in (1.12),  $\Psi_d(\frac{1}{2} - \frac{1}{2^k}) > \frac{1}{2} - \frac{1}{2^k}$  is equivalent to  $v_0^{d-1} < \frac{4}{2^{k+2}}$ , which we aim to show for  $k \geq 4$ . We start with the case  $k \geq 5$ . We have shown in (3.4) that  $v_0^{d-2} \leq 2^{-k}e^{\varepsilon_k}$ , holds, and by an analogous proof,  $v_0^{d-1} \leq 2^{-k}e^{\beta_k}$  holds, where  $\beta_k \equiv \varepsilon_k - \frac{1}{2^{k-1}}(1 - \frac{2(k-1)}{2^k})$ . Thus, it suffices to show that

$$e^{\beta_k} \left(1 + \frac{1}{2^{k-1}}\right) < 4, \quad \text{where} \quad \beta_k \equiv \frac{2(k-1)k \log 2}{2^k} + \frac{4k \log 2 + 2}{2^k} \left(1 - \frac{2(k-1)}{2^k}\right).$$

For  $k = 5$ ,  $e^{\beta_5}(1 + 1/2^4)$  can be computed to arbitrary precision (e.g. by Mathematica), and it can be numerically verified that  $e^{\beta_5}(1 + 1/2^4) < 3.7$ . Further,  $k \rightarrow \beta_k$  is decreasing by comparing  $\beta_k$  and  $\beta_{k+1}$ , thus this concludes the proof for  $k \geq 5$ .

Next, we consider the case  $k = 4$ . Since  $d \rightarrow v_0^{d-1}$  is maximized at  $d = d_{\text{lb}d}(4) \equiv 16.7$ , it suffices to show that  $v_0^{15.7} \leq \frac{2}{9}$  holds, where  $v_0 \equiv \hat{\Psi}_4(1/2 - 1/2^4) = 3410/3753$ . Since  $v_0^{15.7} = (3410/3753)^{15.7}$  can be computed to arbitrary precision (e.g. by Mathematica), it can be checked that  $v_0^{15.7} = (3410/3753)^{15.7} < 0.2221 < \frac{2}{9}$  holds, so this concludes the proof. ◀

**Proof of Lemma 3.1.** By Lemma 3.4,  $\Psi_d(\frac{1}{2} - \frac{1}{2^k}) > \frac{1}{2} - \frac{1}{2^k}$  holds for  $k \geq 4$ . Note that  $\Psi_d(1/2) < 1/2$  holds since  $\hat{\Psi}(x) < 1/2$  holds for any  $x \geq 0$ . Thus, since  $x \rightarrow \Psi_d(x)$  is continuous and differentiable, intermediate value theorem guarantees the existence of the solution to  $\Psi_d(x) = x$  for  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ . Moreover,  $|(\Psi_d)'(x)| < 1$  holds uniformly over  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$  by Lemma 3.3, thus mean value theorem guarantees the uniqueness of the solution to  $\Psi_d(x) = x$  for  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ . ◀

## 3.2 Proof of Lemma 3.2

Recall that  $\star\Phi(d)$  is defined in (1.2) as  $\star\Phi(d) \equiv \Phi(d, x_\star(k, d))$ , where  $x_\star(k, d) \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$  is the solution to  $\Psi_d(x) = x$ , and we defined the function  $\Phi(d, x)$  by

$$\Phi(d, x) \equiv \Phi_k(d, x) := -\log(1-x) - d(1-k^{-1} - d^{-1}) \log(1-2x^k) + (d-1) \log(1-x^{k-1}). \quad (3.6)$$

To prove  $\star\Phi(d_{\text{lb}d}(k)) > 0$  and  $\star\Phi(d_{\text{ub}d}(k)) < 0$ , we show respectively in Lemmas 3.5 and 3.6 that  $\Phi(d_{\text{lb}d}(k), x) > 0$  and  $\Phi(d_{\text{ub}d}(k), x) < 0$  hold uniformly over  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ .

► **Lemma 3.5.** For  $k \geq 4$ ,  $\Phi(d_{\text{lb}d}(k), x) > 0$  holds uniformly over  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ .

**Proof.** Note that rearranging  $\Phi(d, x)$  gives

$$\begin{aligned} \Phi(d, x) &= -\log(1-x) - d((1-k^{-1}) \log(1-2x^k) - \log(1-x^{k-1})) + \log(1-2x^k) - \log(1-x^{k-1}) \\ &\geq -\log(1-x) - d((1-k^{-1}) \log(1-2x^k) - \log(1-x^{k-1})), \end{aligned} \quad (3.7)$$

where the inequality holds since  $\log(1-2x^k) \geq \log(1-x^{k-1})$  holds for  $x \in [0, 1/2]$ . Note that the first term in the right hand side  $x \rightarrow -\log(1-x)$  is convex, so the linear approximation at  $x = 1/2$  shows that  $-\log(1-x) \geq \log 2 + 2(x - 1/2)$  holds. Further, the function  $x \rightarrow (1-k^{-1}) \log(1-2x^k) - \log(1-x^{k-1})$  is increasing since

$$\frac{d}{dx} \left( (1-k^{-1}) \log(1-2x^k) - \log(1-x^{k-1}) \right) = \frac{(k-1)x^{k-2}(1-2x)}{(1-2x^k)(1-x^{k-1})} \geq 0.$$

Thus, the right hand side in (3.7) for  $d = d_{\text{ibd}}(k)$  can further be lower bounded by

$$\begin{aligned}\Phi(d_{\text{ibd}}(k), x) &\geq \log 2 + 2(x - 1/2) + \frac{d_{\text{ibd}}(k)}{k} \cdot \log(1 - 2^{-k+1}) \\ &\geq \log 2 - 2^{-k+1} + \frac{d_{\text{ibd}}(k)}{k} \cdot \log(1 - 2^{-k+1}) =: F(k),\end{aligned}\quad (3.8)$$

where we used  $x \geq 1/2 - 1/2^k$  in the last inequality. Using the inequality  $\log(1 - a) \geq -a - \frac{a^2}{2} - \frac{a^3}{2}$  for  $a = 2^{-k+1} \leq \frac{1}{8}$ , we have, for  $k \geq 5$ , that

$$F(k) = \log 2 - 2^{-k+1} + (2^{k-1} - 2) \log 2 \cdot \log(1 - 2^{-k+1}) \geq \frac{1}{2^k} \left( 3 \log 2 - 2 - \frac{6 \log 2}{2^k} + \frac{8 \log 2}{2^{2k}} \right).$$

For  $k \geq 6$ , the right hand side above is positive since  $3 \log 2 - 2 - \frac{3 \log 2}{32} > 0.01$ , thus (3.8) shows that  $\Phi(d_{\text{ibd}}(k), x) > 0$  holds for  $k \geq 6$  and  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ . For  $k \in \{4, 5\}$ , we can explicitly calculate  $F(k)$  by  $F(4) \equiv \log 2 - 1/8 + (16.7/4) \log(7/8) > 0.01 > 0$ , and  $F(5) \equiv \log 2 - 1/16 + 14 \log 2 \cdot \log(15/16) > 0.004 > 0$ , thus (3.8) again concludes the proof for  $k \in \{4, 5\}$ .  $\blacktriangleleft$

► **Lemma 3.6.** For  $k \geq 4$ ,  $\Phi(d_{\text{ubd}}(k), x) < 0$  holds uniformly over  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ .

**Proof.** We first claim that for  $k \geq 5$ , the function  $x \rightarrow \Phi(d_{\text{ubd}}(k), x)$  is increasing for  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$  and  $d_{\text{ubd}}(k) \equiv 2^{k-1} k \log 2$ . A direct calculation shows that

$$\begin{aligned}\frac{\partial \Phi}{\partial x}(d_{\text{ubd}}(k), x) &= \frac{1}{1-x} - (2^{k-1} k \log 2 - 1)(k-1) \cdot \frac{x^{k-2}(1-2x)}{(1-x^{k-1})(1-2x^k)} - \frac{2x^{k-1}}{1-2x^k} \\ &\geq \frac{1}{\frac{1}{2} + \frac{1}{2^k}} - (2^{k-1} k \log 2 - 1)(k-1) \cdot \frac{x^{k-2}(1-2x)}{(1-x^{k-1})(1-2x^k)} - \frac{4}{2^k - 2},\end{aligned}\quad (3.9)$$

where the inequality holds since  $x \rightarrow (1-x)^{-1}$  increasing, so it is minimized at  $x = 1/2 + 1/2^k$ , and  $x \rightarrow 2x^{k-1}/(1-2x^k)$  is increasing, so it is maximized at  $x = 1/2$ . Further, it is straightforward to check that  $x \rightarrow x^{k-2}(1-2x)$  is decreasing for  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ , thus it is maximized at  $x = 1/2 - 1/2^k$ . Also,  $x \rightarrow (1-x^{k-1})(1-2x^k)$  is minimized at  $x = 1/2$ . Thus, by plugging in these bounds, we can further bound

$$\begin{aligned}\frac{\partial \Phi}{\partial x}(d_{\text{ubd}}(k), x) &\geq 2 - \left( \frac{2}{2^{k-1} + 1} + \frac{4}{2^k - 2} + \frac{(2^{k-1} k \log 2 - 1)(k-1)}{2^{2k-3}} \cdot \left( 1 - \frac{1}{2^{k-1}} \right)^{k-4} \right) \\ &\geq 2 - \left( \frac{2}{2^{k-1} + 1} + \frac{4}{2^k - 2} + \frac{(2^{k-1} k \log 2 - 1)(k-1)}{2^{2k-3}} \right) =: 2 - G(k).\end{aligned}\quad (3.10)$$

Note that the function  $k \rightarrow G(k)$  is increasing for  $k \geq 5$ . Furthermore, using the bound  $\log 2 < 0.7$ , we can bound  $G(5) = \frac{2}{17} + \frac{2}{15} + \frac{80 \log 2 - 1}{32} < 1.97 < 2$ . Therefore,  $\frac{\partial \Phi}{\partial x}(d_{\text{ubd}}(k), x) > 0$  holds for  $k \geq 5$  and  $x \in [\frac{1}{2} - \frac{1}{2^k}, \frac{1}{2}]$ , which proves our first claim.

Consequently, for the case  $k \geq 5$ , it suffices to show that  $\Phi(2^{k-1} k \log 2, x) < 0$  holds for  $x = 1/2$ . A direct calculation gives

$$\Phi\left(2^{k-1} k \log 2, \frac{1}{2}\right) = \log 2 + 2^{k-1} \log 2 \cdot \log\left(1 - \frac{1}{2^{k-1}}\right) < 0,\quad (3.11)$$

where the inequality holds since  $\log(1 - a) < -a$  holds for  $a \in (0, 1)$ . This concludes the proof for  $k \geq 5$ .

It remains to consider the case  $k = 4$ . For  $k = 4$ , we claim that  $x \rightarrow \Phi_4(d_{\text{ubd}}(4), x)$  is convex in the interval  $x \in [\frac{7}{16}, \frac{1}{2}]$ . From the computation of  $\frac{\partial \Phi}{\partial x}(d_{\text{ubd}}(k), x)$  in (3.9), we can calculate the second derivative by

$$\frac{\partial^2 \Phi}{\partial x^2}(d_{\text{ubd}}(4), x) = \frac{d}{dx} \left( \frac{1}{1-x} - \frac{2x^3}{1-2x^4} \right) + 3(32 \log 2 - 1) \cdot \frac{d}{dx} \left( \frac{x^2(2x-1)}{(1-x^3)(1-2x^4)} \right). \quad (3.12)$$

The first term in the right hand side can be bounded by

$$\frac{d}{dx} \left( \frac{1}{1-x} - \frac{2x^3}{1-2x^4} \right) = \frac{1}{(1-x)^2} - \frac{6x^2 + 4x^6}{(1-2x^4)^2} > \frac{1}{(1-\frac{7}{16})^2} - \frac{6(\frac{1}{2})^2 + 4(\frac{1}{2})^6}{(1-2(\frac{1}{2})^4)^2} > 0, \quad (3.13)$$

where the final inequality is equivalent to  $\frac{256}{81} - \frac{100}{49} > 0$ . The second term can be calculated as

$$\frac{d}{dx} \left( \frac{x^2(2x-1)}{(1-x^3)(1-2x^4)} \right) = \frac{x(-16x^8 + 10x^7 + 4x^5 - 4x^4 - x^3 + 6x - 2)}{(1-x^3)^2(1-2x^4)^2}.$$

Note that by neglecting the terms  $10x^7 + 4x^5$  above, we can lower bound

$$-16x^8 + 10x^7 + 4x^5 - 4x^4 - x^3 + 6x - 2 > 6 \cdot \frac{7}{16} - 2 - \left(\frac{1}{2}\right)^3 - 4\left(\frac{1}{2}\right)^4 - 16\left(\frac{1}{2}\right)^8 > 0,$$

thus  $\frac{d}{dx} \left( \frac{x^2(2x-1)}{(1-x^3)(1-2x^4)} \right) > 0$  holds for  $x \in [\frac{7}{16}, \frac{1}{2}]$  as well. Therefore, combining with (3.12) and (3.13) finishes the proof of our claim that  $x \rightarrow \Phi_4(d_{\text{ubd}}(4), x)$  is convex in the interval  $x \in [\frac{7}{16}, \frac{1}{2}]$ .

Thus, by convexity,  $x \rightarrow \Phi_4(d_{\text{ubd}}(4), x)$  is maximized at the end points  $x \in \{7/16, 1/2\}$ , and it suffices to show that  $\Phi_4(d_{\text{ubd}}(4), 7/16) < 0$  and  $\Phi_4(d_{\text{ubd}}(4), 1/2) < 0$ . For  $x = 7/16$ ,  $\Phi_4(d_{\text{ubd}}(4), 7/16)$  can be computed to arbitrary precision (e.g. by Mathematica), and it can be checked that  $\Phi_4(d_{\text{ubd}}(4), 7/16) < -0.08 < 0$ . For  $x = 1/2$ , (3.11) shows that  $\Phi_4(d_{\text{ubd}}(4), 1/2) < 0$  holds. This concludes the proof for the case  $k = 4$ . ◀

**Proof of Lemma 3.2.** By definition,  $\star \Phi(d) = \Phi(d, x_\star(k, d))$  holds, and  $(d, x) \rightarrow \Phi(d, x)$  is clearly continuous. Thus, in order to show the continuity of  $\star \Phi(\cdot)$ , it suffices to show that  $d \rightarrow x_\star(k, d)$  is continuous for any fixed  $k \geq 4$ . To that end, note that the function  $\psi(d, x) := \Psi_d(x) - x$  satisfies  $\frac{\partial \psi}{\partial x} < 0$  by Lemma 3.3. Since  $x_\star(k, d)$  is defined to be the root of  $\psi(d, \cdot)$ , this implies that  $d \rightarrow x_\star(k, d)$  is continuous by the implicit function theorem. As a consequence, we conclude that  $d \rightarrow \star \Phi(d)$  is continuous. Since  $\star \Phi(d_{\text{lb}}(k)) > 0$  holds by Lemma 3.5 and  $\star \Phi(d_{\text{ubd}}(k)) < 0$  holds by Lemma 3.6, we conclude the proof. ◀

---

## References

- 1 Dimitris Achlioptas, Arthur Chtcherba, Gabriel Istrate, and Cristopher Moore. The phase transition in 1-in- $k$  SAT and NAE 3-sat. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 721–722, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=365411.365760>.
- 2 Dimitris Achlioptas and Cristopher Moore. On the 2-colorability of random hypergraphs. In José D. P. Rolim and Salil Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, pages 78–90, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- 3 Dimitris Achlioptas and Cristopher Moore. Random  $k$ -SAT: two moments suffice to cross a sharp threshold. *SIAM J. Comput.*, 36(3):740–762, 2006. doi:10.1137/S0097539703434231.
- 4 Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. *Ann. of Math. (2)*, 162(3):1335–1351, 2005. doi:10.4007/annals.2005.162.1335.

- 5 Dimitris Achlioptas, Assaf Naor, and Yuval Peres. Rigorous location of phase transitions in hard optimization problems. *Nature*, 435(7043):759–764, 2005.
- 6 Dimitris Achlioptas and Yuval Peres. The threshold for random  $k$ -SAT is  $2^k \log 2 - O(k)$ . *J. Amer. Math. Soc.*, 17(4):947–973, 2004. doi:10.1090/S0894-0347-04-00464-3.
- 7 N. Alon and Z. Bregman. Every 8-uniform 8-regular hypergraph is 2-colorable. *Graphs and Combinatorics*, 4(1):303–306, 1988. doi:10.1007/BF01864169.
- 8 Peter Ayre, Amin Coja-Oghlan, Pu Gao, and Noëla Müller. The satisfiability threshold for random linear equations. *Combinatorica*, 40(2):179–235, 2020. doi:10.1007/s00493-019-3897-3.
- 9 Peter Ayre, Amin Coja-Oghlan, and Catherine Greenhill. Lower bounds on the chromatic number of random graphs. *Combinatorica*, 42(5):617–658, 2022. doi:10.1007/s00493-021-4236-z.
- 10 Victor Bapst and Amin Coja-Oghlan. The condensation phase transition in the regular  $k$ -SAT model. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, volume 60 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Art. No. 22, 18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.
- 11 Victor Bapst, Amin Coja-Oghlan, Samuel Hetterich, Felicia Raßmann, and Dan Vilenchik. The condensation phase transition in random graph coloring. *Comm. Math. Phys.*, 341(2):543–606, 2016. doi:10.1007/s00220-015-2464-z.
- 12 Béla Bollobás, Christian Borgs, Jennifer T. Chayes, Jeong Han Kim, and David B. Wilson. The scaling window of the 2-SAT transition. *Random Structures Algorithms*, 18(3):201–256, 2001. doi:10.1002/rsa.1006.
- 13 A. A. Borovkov. Generalization and refinement of the integro-local stone theorem for sums of random vectors. *Theory of Probability & Its Applications*, 61(4):590–612, 2017. doi:10.1137/S0040585X97T988368.
- 14 Evan Chang, Neel Kolhe, and Youngtak Sohn. Upper bounds on the 2-colorability threshold of random  $d$ -regular  $k$ -uniform hypergraphs for  $k \geq 3$ . *arXiv:2308.02075*, 2023.
- 15 V. Chvatal and B. Reed. Mick gets some (the odds are on his side) (satisfiability). In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, SFCS '92, pages 620–627, Washington, DC, USA, 1992. IEEE Computer Society. doi:10.1109/SFCS.1992.267789.
- 16 Amin Coja-Oghlan. Upper-bounding the  $k$ -colorability threshold by counting covers. *Electron. J. Combin.*, 20(3):Paper 32, 28, 2013.
- 17 Amin Coja-Oghlan, Charilaos Efthymiou, and Samuel Hetterich. On the chromatic number of random regular graphs. *J. Combin. Theory Ser. B*, 116:367–439, 2016. doi:10.1016/j.jctb.2015.09.006.
- 18 Amin Coja-Oghlan, Florent Krzakala, Will Perkins, and Lenka Zdeborová. Information-theoretic thresholds from the cavity method. *Adv. Math.*, 333:694–795, 2018. doi:10.1016/j.aim.2018.05.029.
- 19 Amin Coja-Oghlan and Konstantinos Panagiotou. Catching the  $k$ -NAESAT threshold [extended abstract]. In *STOC'12—Proceedings of the 2012 ACM Symposium on Theory of Computing*, pages 899–907. ACM, New York, 2012. doi:10.1145/2213977.2214058.
- 20 Amin Coja-Oghlan and Konstantinos Panagiotou. Going after the  $k$ -sat threshold. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 705–714, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488698.
- 21 Amin Coja-Oghlan and Konstantinos Panagiotou. The asymptotic  $k$ -SAT threshold. *Adv. Math.*, 288:985–1068, 2016. doi:10.1016/j.aim.2015.11.007.
- 22 Amin Coja-Oghlan and Will Perkins. Spin systems on Bethe lattices. *Communications in Mathematical Physics*, 372(2):441–523, 2019. doi:10.1007/s00220-019-03544-y.
- 23 Amin Coja-Oghlan and Dan Vilenchik. Chasing the  $k$ -colorability threshold. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science—FOCS '13*, pages 380–389. IEEE Computer Soc., Los Alamitos, CA, 2013. doi:10.1109/FOCS.2013.48.

- 24 Amin Coja-Oghlan and Lenka Zdeborová. The condensation transition in random hypergraph 2-coloring. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 241–250. ACM, New York, 2012.
- 25 L. Dall'Asta, A. Ramezanpour, and R. Zecchina. Entropy landscape and non-gibbs solutions in constraint satisfaction problems. *Physical Review E*, 77(3), March 2008. doi:10.1103/physreve.77.031118.
- 26 Amir Dembo and Ofer Zeitouni. *Large deviations techniques and applications*, volume 38 of *Stochastic Modelling and Applied Probability*. Springer-Verlag, Berlin, 2010. doi:10.1007/978-3-642-03311-7.
- 27 Martin Dietzfelbinger, Andreas Goerdts, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight thresholds for cuckoo hashing via XORSAT. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 213–225, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- 28 Jian Ding, Allan Sly, and Nike Sun. Satisfiability threshold for random regular nae-sat. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 814–822, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591862.
- 29 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large  $k$ . In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 59–68, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746619.
- 30 Jian Ding, Allan Sly, and Nike Sun. Maximum independent sets on random regular graphs. *Acta Math.*, 217(2):263–340, 2016. doi:10.1007/s11511-017-0145-9.
- 31 Jian Ding, Allan Sly, and Nike Sun. Satisfiability threshold for random regular NAE-SAT. *Commun. Math. Phys.*, 341(2):435–489, 2016.
- 32 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large  $k$ . *Annals of Mathematics*, 196(1):1–388, 2022. doi:10.4007/annals.2022.196.1.1.
- 33 Olivier Dubois and Jacques Mandler. The 3-XORSAT threshold. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS '02, pages 769–778, Washington, DC, USA, 2002. IEEE Computer Society. URL: <http://dl.acm.org/citation.cfm?id=645413.652160>.
- 34 Martin Dyer, Alan Frieze, and Catherine Greenhill. On the chromatic number of a random hypergraph. *Journal of Combinatorial Theory, Series B*, 113:68–122, 2015. doi:10.1016/j.jctb.2015.01.002.
- 35 Silvio Franz and Michele Leone. Replica bounds for optimization problems and diluted spin systems. *Journal of Statistical Physics*, 111(3):535–564, 2003. doi:10.1023/A:1022885828956.
- 36 Yuzhou Gu and Yury Polyanskiy. Uniqueness of bp fixed point for the potts model and applications to community detection. *arXiv preprint, arXiv:2303.14688*, 2023.
- 37 Francesco Guerra. Broken replica symmetry bounds in the mean field spin glass model. *Communications in Mathematical Physics*, 233(1):1–12, 2003. doi:10.1007/s00220-002-0773-5.
- 38 Michael A. Henning and Anders Yeo. 2-colorings in  $k$ -regular  $k$ -uniform hypergraphs. *European Journal of Combinatorics*, 34(7):1192–1202, 2013. doi:10.1016/j.ejc.2013.04.005.
- 39 Michael A. Henning and Anders Yeo. Not-all-equal 3-sat and 2-colorings of 4-regular 4-uniform hypergraphs. *Discrete Mathematics*, 341(8):2285–2292, 2018. doi:10.1016/j.disc.2018.05.002.
- 40 Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000. doi:10.1002/9781118032718.
- 41 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. doi:10.1007/978-1-4684-2001-2\_9.
- 42 Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures Algorithms*, 12(3):253–269, 1998. doi:10.1002/(SICI)1098-2418(199805)12:3<253::AID-RSA3>3.3.CO;2-H.




- 43 Florent Krzakala, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007. doi:10.1073/pnas.0703685104.
- 44 Marc Lelarge and Mendes Oulamara. Replica bounds by combinatorial interpolation for diluted spin systems. *Journal of Statistical Physics*, 173(3–4):917–940, February 2018. doi:10.1007/s10955-018-1964-6.
- 45 Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-sat from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006. doi:10.1002/rsa.20090.
- 46 M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002. doi:10.1126/science.1073287.
- 47 Marc Mézard and Andrea Montanari. *Information, physics, and computation*. Oxford Graduate Texts. Oxford University Press, Oxford, 2009. doi:10.1093/acprof:oso/9780198570837.001.0001.
- 48 Andrea Montanari, Feng Ruan, Youngtak Sohn, and Jun Yan. The generalization error of max-margin linear classifiers: Benign overfitting and high-dimensional asymptotics in the overparametrized regime. *arXiv*, 2019. arXiv:1911.01544.
- 49 Danny Nam, Allan Sly, and Youngtak Sohn. One-step replica symmetry breaking of random regular NAE-SAT I. *arXiv preprint*, 2020. arXiv:2011.14270.
- 50 Danny Nam, Allan Sly, and Youngtak Sohn. One-step replica symmetry breaking of random regular NAE-SAT. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 310–318, 2022. doi:10.1109/FOCS52979.2021.00039.
- 51 Danny Nam, Allan Sly, and Youngtak Sohn. One-step replica symmetry breaking of random regular NAE-SAT II. *Communications in Mathematical Physics*, 405(3):61, 2024. doi:10.1007/s00220-023-04868-6.
- 52 Dmitry Panchenko and Michel Talagrand. Bounds for diluted mean-fields spin glass models. *Probability Theory and Related Fields*, 130(3):319–336, 2004. doi:10.1007/s00440-004-0342-2.
- 53 Boris Pittel and Gregory B. Sorkin. The satisfiability threshold for  $k$ -XORSAT. *Combin. Probab. Comput.*, 25(2):236–268, 2016. doi:10.1017/S0963548315000097.
- 54 P. D. Seymour. On the two-coloring of hypergraphs. *The Quarterly Journal of Mathematics*, 25(1):303–311, January 1974. doi:10.1093/qmath/25.1.303.
- 55 Mariya Shcherbina and Brunello Tirozzi. Rigorous solution of the Gardner problem. *Communications in Mathematical Physics*, 234(3):383–422, 2003.
- 56 Allan Sly and Youngtak Sohn. Local geometry of NAE-SAT solutions in the condensation regime. *arXiv preprint*, 2023. arXiv:2305.17334.
- 57 Allan Sly, Nike Sun, and Yumeng Zhang. The number of solutions for random regular NAE-SAT. In *Proceedings of the 57th Symposium on Foundations of Computer Science*, FOCS '16, pages 724–731, 2016.
- 58 Allan Sly, Nike Sun, and Yumeng Zhang. The number of solutions for random regular NAE-SAT. *Probability Theory and Related Fields*, 182(1-2):1–109, 2022. doi:10.1007/s00440-021-01029-5.
- 59 Michel Talagrand. *Mean Field Models for Spin Glasses: Volume I*. Springer-Verlag, Berlin, 2010.
- 60 Qian Yu and Yury Polyanskiy. Ising model on locally tree-like graphs: Uniqueness of solutions to cavity equations. *arXiv preprint*, 2022. arXiv:2211.15242.



# Improved Bounds for High-Dimensional Equivalence and Product Testing Using Subcube Queries

Tomer Adar  

Technion – Israel Institute of Technology, Haifa, Israel

Eldar Fischer 

Technion – Israel Institute of Technology, Haifa, Israel

Amit Levi  

University of Haifa, Israel

---

## Abstract

We study property testing in the subcube conditional model introduced by Bhattacharyya and Chakraborty (2017). We obtain the first equivalence test for  $n$ -dimensional distributions that is quasi-linear in  $n$ , improving the previously known  $\tilde{O}(n^2/\varepsilon^2)$  query complexity bound to  $\tilde{O}(n/\varepsilon^2)$ . We extend this result to general finite alphabets with logarithmic cost in the alphabet size.

By exploiting the specific structure of the queries that we use (which are more restrictive than general subcube queries), we obtain a cubic improvement over the best known test for distributions over  $\{1, \dots, N\}$  under the interval querying model of Canonne, Ron and Servedio (2015), attaining a query complexity of  $\tilde{O}((\log N)/\varepsilon^2)$ , which for fixed  $\varepsilon$  almost matches the known lower bound of  $\Omega((\log N)/\log \log N)$ . We also derive a product test for  $n$ -dimensional distributions with  $\tilde{O}(n/\varepsilon^2)$  queries, and provide an  $\Omega(\sqrt{n}/\varepsilon^2)$  lower bound for this property.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Distribution testing, conditional sampling, sub-cube sampling

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.48

**Category** RANDOM

**Funding** *Eldar Fischer*: Research supported by an Israel Science Foundation grant number 879/22.

## 1 Introduction

Property testing seeks to efficiently distinguish between objects that have some property and objects that are  $\varepsilon$ -far from any object that has it, with respect to a predefined metric and a proximity parameter  $\varepsilon$ . Property testing of functions, and in particular string testing, was initiated in [18, 6]. The term “efficiently” usually refers to a sublinear amount of resources for moderately-sized tasks, but for high-dimensional inputs practicality mandates to restrict this amount further to the poly-logarithmic scale.

The study of distribution testing was implicitly initiated in [14] (motivated by a problem in graph testing), and formally defined as the *sampling model* in [2, 1]. In this model, the algorithm gets access to a sequence of independent unconditional samples from the input distribution, and then decides whether to accept or reject based on them. A major topic of investigation concerns testing that a distribution over  $\{1, \dots, n\}$  is the uniform one. A long string of results, starting with the original [14] and culminating in [17], has reached the tight bound of  $\Theta(\sqrt{n}/\varepsilon^2)$ .

A square-root lower bound on one of the most basic of properties is impractical in many real-world settings. For example, the square-root sample complexity of uniformity testing is impractical when the input ranges over 100-bit binary strings, an example which is still



© Tomer Adar, Eldar Fischer, and Amit Levi;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 48; pp. 48:1–48:21



Leibniz International Proceedings in Informatics

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

smaller than real-world inputs. There are three approaches to overcome this problem: the first, on which we expand below, is moving to stronger query models. The second is narrowing the scope of the allowable inputs, such as *product distributions* and *Bayesian networks* [8]. The third is moving to a model where the metric is more lax (usually coupled with an even weaker query model), such as the Huge Object Model defined in [15] that is concerned with the earth-mover distance metric.

We focus here on the first (and most common) approach to the scaling problem, that of considering a model with stronger queries. In the distribution testing setting, this usually focuses on *conditional sampling*. Instead of drawing a sequence of independent unconditional samples, we allow the algorithm to choose a subset of the possible outcomes and draw a sample conditioned on belonging to it. The models within this paradigm differ in the subset conditions they allow.

The first investigation of such a model was the fully conditional model [10, 9]. In this model, the algorithm can choose *any* subset to condition on. This model is very powerful but not realistic. If we are able to restrict the input distribution to any subset, we probably already have access to an explicit description of the distribution and thus have no need to sample it.

Further studies consider restricted forms of the conditional model. For example, [9] considers two additional models: the pair model and the interval model. In the pair model, the algorithm can still draw unconditional samples, and additionally it can draw conditional samples from any subset of two elements. Uniformity testing in the pair model requires  $\tilde{\Theta}(1/\varepsilon^2)$  samples. In the interval model, we test distributions over  $\Omega = \{1, \dots, N\}$  (for some  $N$ ), and the algorithm can condition on interval sets, which are sets of the form  $\{a \leq x \leq b\}$ . The lower bound for uniformity testing in the interval model is  $\tilde{\Omega}(\log N)$ . The best known upper bound is  $\tilde{O}((\log N)^3/\varepsilon^2)$ .

The subcube conditional model [4] is motivated by database analysis. In this model, we test distributions over the set  $\prod_{i=1}^n \Omega_i$ , and the algorithm can query *subcube subsets*, which are sets of the form  $\prod_{i=1}^n A_i$  where  $A_i \subseteq \Omega_i$  for every  $1 \leq i \leq n$ . While not being extremely restrictive, its queries correspond to selection by attribute values, which is common in practice. Some of the prior work refers to a weaker variant of this model, where each  $A_i$  is either trivial ( $A_i = \Omega_i$ ) or a singleton ( $A_i = \{a_i\}$  for some  $a_i \in \Omega_i$ ). In this paper we mainly deal with the strong model, where general  $A_i \subseteq \Omega_i$  are allowed. In the most investigated setting, the binary setting where  $\Omega_i = \{0, 1\}$  for all  $i$ , the two models are the same.

Uniformity testing [7, 12] can be done in the weak model using  $\tilde{O}(m^{21}\sqrt{n}/\varepsilon^2)$  queries where  $m = \max_i |\Omega_i|$ , and requires at least  $\Omega(\sqrt{mn}/\varepsilon^2)$  queries [3]. Other properties studied under the subcube model include identity to some fixed distribution [4, 5] and having a probability density function supported on a low-dimensional subspace [11].

Two related properties are of particular interest in distribution testing, the identity property and the equivalence property. In the identity property, a *reference distribution* is given to the algorithm in advance, and the task is to check whether the input distribution is identical to it. The equivalence property (henceforth: EQUIVALENCE) has an input consisting of *two* distributions, both of which accessible through the testing model, and the task is to check whether they are equal to each other.

In the subcube conditional model, since the input distribution is defined over a set of tuples, another natural property is that of being a product distribution (henceforth: PRODUCT). A distribution over tuples is called a product if its entries are independently distributed.

Our main upper bound result is a test for EQUIVALENCE that for the binary setting ( $\Omega_i = \{0, 1\}$ ) uses only  $\tilde{O}(n/\varepsilon^2)$  subcube queries, which improves on the previously known result of  $\tilde{O}(n^2/\varepsilon^2)$  from [4]. Additionally our result uses only *prefix queries* from one distribution and *marginal prefix queries* from the other, which we define below and are rather restricted forms of subcube queries. One can think of prefix queries as the queries that can be made fast when the database is sorted according to a concatenation of its attributes in a pre-defined order (functioning as its primary key). Importantly, the use of restricted queries allows us to derive an improved test also for PRODUCT, and to generalize the test to general  $\Omega_1, \dots, \Omega_n$  with a logarithmic cost in the alphabet size.

The restricted form of our queries also allows us to tighten the previously known upper bound on equivalence testing (and through it the special case of uniformity testing) in the interval conditional model, obtaining an upper bound of  $\tilde{O}(\log N/\varepsilon^2)$  interval queries, which matches the lower bound for every fixed  $\varepsilon$  up to poly-double-logarithmic factors in  $N$ .

We complement our upper bound for PRODUCT with an  $\Omega(\sqrt{n}/\varepsilon^2)$  lower bound. The question of whether we can go below  $O(n)$  for testing our properties (even for the binary setting and a fixed  $\varepsilon$ ) remains open.

## 2 Organization of the paper

In Section 3 we summarize our contributions. After some preliminaries in Section 4, we provide the core of our main proofs, followed by the more technical details. In Section 5 we provide the EQUIVALENCE testing upper bound in the binary setting, and a short proof of the corollary about interval queries. In Section 6 we provide the PRODUCT testing lower bound.

The technical part of the paper follows. In Section 7 we prove the technical lemmas whose proofs were deferred from Section 5. Then we prove the theorems derived from this upper bound: in Section 8 we extend the EQUIVALENCE test to the non-binary setting, and in Section 9 we derive a test for PRODUCT. In Section 10 we prove the technical lemmas from Section 6.

All upper bound proofs implicitly construct their algorithms. For reference, explicit representations of the binary setting algorithms for EQUIVALENCE and PRODUCT are given in the appendix.

## 3 Our results

We improve on the previously known result of  $\tilde{O}(n^2/\varepsilon^2)$  queries for equivalence testing of two distributions over  $\{0, 1\}^n$  [4]. Our methods can also supersede the  $\tilde{O}(n/\varepsilon)$  algorithm of [5] which tests identity with a distribution given in advance that belongs to a very restricted class of inputs (their parameter refers to KL-divergence, which indeed incurs a quadratic gap when converted to total-variation distance). We provide more details on the latter below (before Lemma 25).

► **Theorem 20.** *Let  $\tau, \mu$  be two distributions over  $\{0, 1\}^n$ , where  $\tau$  is accessible through the prefix oracle access and  $\mu$  is accessible through the marginal prefix oracle access. For every  $\varepsilon > 0$  we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}(n/\varepsilon^2)$  queries.*

The prefix queries in the statement above are a special case of subcube queries, but they can also be seen as interval queries, allowing us to prove the following corollary:

► **Corollary 21.** *Let  $\tau, \mu$  be two distributions over  $\{1, \dots, N\}$ , both accessible through the interval oracle. Then we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}((\log N)/\varepsilon^2)$  queries.*

We show a lower bound for testing a distribution  $\mu$  over  $\{0, 1\}^n$  for being a product. To the best of our knowledge, it is the first lower bound for product testing in the binary setting. Our construction is similar to the lower bound for uniformity testing of [8].

► **Theorem 26.** *Every  $\varepsilon$ -test for PRODUCT must make at least  $\Omega(\sqrt{n}/\varepsilon^2)$  subcube queries.*

We generalize our upper bound for equivalence testing to strings of size  $n$  over larger alphabets. Our result is incomparable with the previously known result of  $\tilde{O}(\frac{n^5}{\varepsilon^5} \log \log |\Omega|)$  [4]. Note that the cited result refers to strings over a single alphabet (that is,  $\Omega^n$ ), whereas our result refers to strings over mixed alphabets (that is,  $\prod_{i=1}^n \Omega_i$ ). Additionally, our result is more efficient when  $|\Omega|$  is not very large with respect to  $n$ .

► **Theorem 40.** *Let  $\mu$  and  $\tau$  be two distributions over  $\prod_{i=1}^n \Omega_i$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. If  $\mu$  is accessible through the marginal prefix oracle and  $\tau$  is accessible through the prefix oracle, then we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}(\sum_{i=1}^n \log_2 |\Omega_i|/\varepsilon^2)$  queries.*

We apply the same generalization for product testing as well. In the binary setting, it also improves on the previously known result of  $\tilde{O}(n^2/\varepsilon^2)$  [4].

► **Theorem 45.** *Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ . For every  $0 < \varepsilon < 1$ , we can distinguish between the case where  $\mu$  is a product distribution and the case where it is  $\varepsilon$ -far from every product distribution at the cost of  $\tilde{O}(\sum_{i=1}^n \log |\Omega_i|/\varepsilon^2)$  subcube queries. Moreover, if  $|\Omega_i| = 2$  for every  $1 \leq i \leq n$ , then all these queries are prefix queries.*

## 4 Preliminaries

For brevity, for  $m \in \mathbb{N}$  we let  $[m]$  denote the set  $\{1, \dots, m\}$ .

► **Definition 1** (Bernoulli distribution). *Let  $0 \leq p \leq 1$ . The Bernoulli distribution with parameter  $p$ , denoted by  $\text{Ber}(p)$ , is the distribution over  $\{0, 1\}$  whose probability to draw 1 is  $p$ .*

► **Definition 2** (Conditional distribution). *Let  $\mu$  be a distribution over  $\Omega$ , and let  $B \subseteq \Omega$  be an event. The corresponding conditional distribution is denoted by  $\mu|_B$  and is defined by  $\mu|_B(x) = 0$  for  $x \notin B$  and  $\mu|_B(x) = \mu(x)/\Pr_\mu[B]$  for  $x \in B$ .*

► **Definition 3** (Index-restricted distribution). *Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ , and let  $I \subseteq [n]$  be a set of indices. We use  $\mu|_I$  to denote the distribution that draws  $x \sim \mu$ , and returns the restricted string  $x|_I \in \prod_{i \in I} \Omega_i$ .*

Note that the correct way to parse the overloaded notation  $\mu|_I^B$  is “ $(\mu|_B)|_I$ ”, that is, we first apply the condition and then restrict the indices.

► **Definition 4** (Common statistical divergence measures). *Let  $\mu$  and  $\tau$  be two distributions over a finite set  $\Omega$ . We use two well known divergence measures, namely the total-variation distance  $d_{\text{TV}}(\mu, \tau) = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \tau(x)| = \max_{E \subseteq \Omega} |\mu(E) - \tau(E)|$  and the KL-divergence  $D_{\text{KL}}(\mu, \tau) = \mathbb{E}_{x \sim \mu} \left[ \log_2 \frac{\mu(x)}{\tau(x)} \right]$ .*

Here we define the query model for distributions over binary strings. The definitions for general alphabets appear in Section 8.

► **Definition 5** (Subcube oracle access). *Let  $\mu$  be an unknown distribution over  $\{0, 1\}^n$ . The subcube oracle has a set  $I \subseteq [n]$  and a string  $w \in \{0, 1\}^I$  as input, and its output distributes as  $\mu|^{x_I=w}$ . For technical reasons, if  $\Pr_\mu[x_I = w] = 0$ , then the oracle indicates an error. Note that the answers of the oracle are fully independent of the answers that were given to previous queries.*

We note that the “error behavior” in the above definition does not really affect our results, since all our upper bounds use only restrictions to guaranteed positive probability outcomes, while our lower bounds use distributions that have no zero-probability elements.

► **Definition 6** (Prefix oracle access). *Let  $\mu$  be an unknown distribution over  $\{0, 1\}^n$ . The prefix oracle is a restricted case of the subcube oracle, where  $I = [k]$  for some  $0 \leq k \leq n - 1$ .*

► **Definition 7** (Marginal subcube oracle access). *Let  $\mu$  be an unknown distribution over  $\{0, 1\}^n$ . The marginal subcube oracle has a set  $I \subseteq [n]$ , an index  $i \in [n] \setminus I$  and a string  $w \in \{0, 1\}^I$  as input, and its output is a single bit that distributes as  $\mu|_i^{x_I=w}$ . For technical reasons, if  $\Pr_\mu[x_I = w] = 0$ , then the oracle indicates an error.*

► **Definition 8** (Marginal prefix oracle access). *Let  $\mu$  be an unknown distribution over  $\{0, 1\}^n$ . The marginal prefix oracle is a restricted case of the marginal subcube oracle, where  $I = [i - 1]$ .*

We now define the interval querying model for which we derive a new bound.

► **Definition 9** (Interval oracle access). *Let  $\mu$  be an unknown distribution over  $[N]$ . The interval oracle has two elements  $1 \leq a \leq b \leq N$  as input, and its output distributes as  $\mu|_{\{a, \dots, b\}}$ .*

We next define our properties for distributions over general alphabet strings, and what it means to test for a property.

► **Definition 10** (EQUIVALENCE). *A pair of distributions  $\mu, \tau$  over  $\prod_{i=1}^n \Omega_i$  belongs to the equivalence property if  $\mu = \tau$ . The distance of a given pair  $(\mu, \tau)$  from EQUIVALENCE is explicitly defined as  $d_{\text{TV}}(\mu, \tau)$  instead of the natural  $\inf_\nu (d_{\text{TV}}(\mu, \nu) + d_{\text{TV}}(\tau, \nu))$ . The two quantities are easily seen to be identical using the triangle inequality.*

► **Definition 11** (PRODUCT). *A distribution  $\mu$  over  $\prod_{i=1}^n \Omega_i$  is called a product distribution if there exist distributions  $\mu_1, \dots, \mu_n$  over  $\Omega_1, \dots, \Omega_n$  respectively, for which  $\mu \sim \mu_1 \times \dots \times \mu_n$ . We denote by PRODUCT the set of all product distributions over  $\prod_{i=1}^n \Omega_i$ .*

► **Definition 12** ( $\varepsilon$ -test). *For some property  $\mathcal{P}$  of distributions (which is a set of distributions) and  $\varepsilon > 0$ , we say that an algorithm  $\mathcal{A}$  is an  $\varepsilon$ -test for  $\mathcal{P}$  if:*

- For every input distribution  $\mu \in \mathcal{P}$ ,  $\mathcal{A}$  accepts with probability at least  $\frac{2}{3}$ .
- For every input distribution  $\mu$  for which  $d_{\text{TV}}(\mu, \nu) > \varepsilon$  for all  $\nu \in \mathcal{P}$ ,  $\mathcal{A}$  rejects with probability at least  $\frac{2}{3}$ .

## 5 Linear algorithm for Equivalence

To construct our improvement on the algorithm of [4], we first define some templates for analyzing and comparing differences between distributions.

► **Definition 13** (Single-bit divergence). We call  $d : [0, 1] \times [0, 1] \rightarrow [0, \infty)$  a single-bit divergence if:

- For every  $p, q \in [0, 1]$ ,  $d(p, q) = 0$  if and only if  $p = q$  (Positivity).
- For every  $p' \leq p \leq q \leq q'$ ,  $d(p, q) \leq d(p', q')$  and  $d(q, p) \leq d(q', p')$  (Monotonicity).

Three useful single-bit divergences are  $\text{TV}(p, q) = |p - q|$ ,  $\text{KL}(p, q) = p \log_2 \frac{p}{q} + (1 - p) \log_2 \frac{1-p}{1-q}$  and  $\chi^2(p, q) = \frac{(p-q)^2}{(p+q)(2-(p+q))}$ . Note that all three are derived from their probability-theoretic counterparts for  $\text{Ber}(p)$  and  $\text{Ber}(q)$ . We have two motivations to prefer this non-standard form of  $\chi^2$ : first, the symmetry matches the idea of two unknown distributions, which is not the case in standard  $\chi^2$ -tests, and second, it is bounded by 1, which makes the analysis more similar to total-variation distance than to KL-divergence.

► **Definition 14** (Slice-wise divergence). Let  $d$  be a single-bit divergence. For every  $n$  and two distributions  $\tau, \mu$  over  $\{0, 1\}^n$ , the slice-wise divergence of  $\tau$  and  $\mu$  with respect to  $d$  is:

$$\Delta_d(\tau, \mu) = \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [d(\tau|_i^{x_{[i-1]}=w_{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w_{[i-1]}}(1))]$$

(note that the  $d$ -divergence is fed the probabilities of the two single-bit distributions to draw 1).

Recall the  $\tilde{O}(n^5/\text{poly}(\varepsilon))$  algorithm of [4], which is actually  $\tilde{O}(n^2/\varepsilon^2)$  for the binary setting. As a motivation, the paper uses (and has a self-contained proof of) the following bound:

$$d_{\text{TV}}(\tau, \mu) \leq \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [d_{\text{TV}}(\tau|_i^{x_{[i-1]}=w_{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w_{[i-1]}}(1))] = \Delta_{\text{TV}}(\tau, \mu)$$

The algorithm then distinguishes between  $\Delta_{\text{TV}}(\tau, \mu) = 0$  (which always holds if  $\tau = \mu$ ) and  $\Delta_{\text{TV}}(\tau, \mu) > \varepsilon$  (which always holds if  $d_{\text{TV}}(\tau, \mu) > \varepsilon$ ).

To improve their test, inside the slice-wise divergence expression we substitute the single-bit total-variation distance with the single-bit  $\chi^2$ -distance, and analyze the more convenient  $\Delta_{\chi^2}(\tau, \mu)$ .

The following is immediate, and in fact holds for every slice-wise divergence:

► **Observation 15.** Let  $\tau, \mu$  be two distributions over  $\{0, 1\}^n$ . If  $\tau = \mu$  then  $\Delta_{\chi^2}(\tau, \mu) = 0$ .

The following lemma, which we prove in Subsection 5.1, provides the conversion from the total variation distance to the slice-wise chi-square divergence.

► **Lemma 16.** Let  $\tau, \mu$  be two distributions over  $\{0, 1\}^n$ . If  $\tau \neq \mu$ , then  $\Delta_{\chi^2}(\tau, \mu) \geq \frac{(d_{\text{TV}}(\tau, \mu))^2}{24 \log(2n/d_{\text{TV}}(\tau, \mu))}$ .

The following lemma, which we prove in Section 7, states that we can distinguish between single bit distributions using linearly many samples with respect to the inverse of their  $\chi^2$ -divergence.

► **Lemma 17.** Let  $p, q \in [0, 1]$  be two probabilities. Given unconditional sampling access to  $\text{Ber}(p)$  and  $\text{Ber}(q)$ , we can distinguish, with probability  $\frac{2}{3}$ , between the case where  $p = q$  and the case where  $\chi^2(p, q) > \varepsilon$ , at the cost of  $O(1/\varepsilon)$  samples from each of them.

We use a common variant of the Levin's work balance method:



► **Lemma 18** ([16], optimization exercise in [13]). *Let  $X$  be a non-negative random variable that is bounded by 1. Assume that there exists some random variable  $Y$  such that for every  $y \in \text{supp}(Y)$  and every  $\varepsilon' > 0$ , we can distinguish between  $\mathbb{E}[X|Y = y] = 0$  and  $\mathbb{E}[X|Y = y] > \varepsilon'$  using some black-box algorithm whose resource cost is  $O(1/\varepsilon')$ . Also, assume that we can draw independent unconditional samples from  $Y$  at resource cost  $O(1)$  per sample. Then we can distinguish between  $\mathbb{E}[X] = 0$  and  $\mathbb{E}[X] > \varepsilon$  at a total resource cost of  $O(\varepsilon^{-1} \log^2(1/\varepsilon))$ .*

Using the above we can efficiently detect a large slicewise  $\chi^2$  divergence between two distributions.

► **Lemma 19.** *Let  $\tau, \mu$  be two distributions over  $\{0, 1\}^n$ . Then for every  $\rho > 0$ , we can distinguish between  $\Delta_{\chi^2}(\tau, \mu) = 0$  and  $\Delta_{\chi^2}(\tau, \mu) > \rho$  using  $O(\frac{n}{\rho} \log^2 \frac{n}{\rho})$  prefix queries.*

**Proof.** We normalize the divergence:

$$\begin{aligned} \frac{1}{n} \Delta_{\chi^2}(\tau, \mu) &= \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\chi^2(\tau|_i^{x_{[i-1]}=w_{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w_{[i-1]}}(1))] \\ &= \mathbb{E}_{\substack{w \sim \tau \\ i \sim [n]}} [\chi^2(\tau|_i^{x_{[i-1]}=w_{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w_{[i-1]}}(1))] \end{aligned}$$

Then we apply Levin's method, Lemma 18, with the following input.  $Y$  is a random variable that receives a value  $(i, w)$  where  $i$  is uniformly drawn from  $[n]$  and  $w$  is drawn from  $\tau$  (independently of  $i$ ). The parameter  $\varepsilon$  is set to  $\rho/n$ , and the random variable  $X$  is defined as a function of  $Y = (i, w)$  (meaning that it is constant when conditioned on a specific value of  $Y$ ) by

$$X(i, w) = \chi^2(\mu|_i^{x_{[i-1]}=w_{[i-1]}}(1), \tau|_i^{x_{[i-1]}=w_{[i-1]}}(1))$$

The distinction between the cases  $X(Y) = 0$  and  $X(Y) > \varepsilon'$  is performed using  $O(1/\varepsilon')$  many queries (our resource cost) through the single-bit  $\chi^2$ -test of Lemma 17. ◀

Our main theorem follows immediately from the above statements:

► **Theorem 20.** *Let  $\tau, \mu$  be two distributions over  $\{0, 1\}^n$ , where  $\tau$  is accessible through the prefix oracle access and  $\mu$  is accessible through the marginal prefix oracle access. For every  $\varepsilon > 0$  we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}(n/\varepsilon^2)$  queries.*

**Proof.** Apply Lemma 19 using  $\rho = \frac{\varepsilon^2}{24 \log(2n/\varepsilon)}$ . Completeness follows from Observation 15 and soundness follows from Lemma 16. ◀

► **Corollary 21.** *Let  $\tau, \mu$  be two distributions over  $\{1, \dots, N\}$ , both accessible through the interval oracle. Then we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}((\log N)/\varepsilon^2)$  queries.*

**Proof.** Without loss of generality, assume that  $N = 2^\ell$  for some integer  $\ell$  (otherwise we just pad the two distributions with zero-probability elements). For every  $i \geq 0$ , let  $\text{bin}_i : \{0, \dots, 2^i - 1\} \rightarrow \{0, 1\}^i$  and  $\text{unbin}_i : \{0, 1\}^i \rightarrow \{0, \dots, 2^i - 1\}$  be the mappings between small integers and their representation as  $i$ -bit strings. That is,  $\text{unbin}_i(x_1, \dots, x_i) = \sum_{j=1}^i 2^{i-j} x_j$  and  $\text{bin}_i = (\text{unbin}_i)^{-1}$ .

To apply our equivalence algorithm on distributions over  $[N]$ , every  $t \in [N]$  is interpreted as an  $\ell$ -bit string (using the map  $t \rightarrow \text{bin}_\ell(t - 1)$ ). Then every prefix query  $\tau|^{x_{[i-1]}=w}$  (respectively  $\mu|^{x_{[i-1]}=w}$ ) is simulated using the interval query  $\tau|^{a, \dots, b}$  (respectively  $\mu|^{a, \dots, b}$ ), where  $a = 2^{N-i} \text{unbin}_{i-1}(w) + 1$  and  $b = 2^{N-i} (\text{unbin}_{i-1}(w) + 1)$ . Since the simulated algorithm uses  $\tilde{O}(\ell/\varepsilon^2)$  prefix queries, the simulation uses  $\tilde{O}((\log N)/\varepsilon^2)$  interval queries. ◀

## 5.1 Proof of Lemma 16

The proof works by comparing the TV-distance and the slice-wise chi-square divergence with the KL-divergence, which is equal to its slice-wise version as per the following lemmas.

► **Lemma 22** (Basic chain rule for  $D_{\text{KL}}$ , folklore). *Let  $\mu$  and  $\tau$  be distributions over  $\Omega$ . Then for every random variable  $X : \Omega \rightarrow R$ , we have the equality  $D_{\text{KL}}(\mu, \tau) = D_{\text{KL}}(X(\mu), X(\tau)) + \mathbb{E}_{x \sim X(\tau)} [D_{\text{KL}}(\mu|^{X=x}, \tau|^{X=x})]$ .*

► **Lemma 23** (Chain rule for  $D_{\text{KL}}$ , repeated form). *Let  $\mu$  and  $\tau$  be distributions over  $\{0, 1\}^n$ . Then  $D_{\text{KL}}(\mu, \tau) = \sum_{i=1}^n \mathbb{E}_{w \sim \mu} [D_{\text{KL}}(\mu|_i^{x_{[i-1]}=w^{[i-1]}}, \tau|_i^{x_{[i-1]}=w^{[i-1]}})] = \Delta_{\text{KL}}(\mu, \tau)$ .*

**Proof.** We obtain this result by repeatedly applying the basic chain rule using induction. ◀

The following technical lemma, proved in Section 7, provides some connection between the chi-square distance and the KL-divergence of Bernoulli distributions.

► **Lemma 24.** *Let  $p, q \in [0, 1]$ . Then  $\chi^2(p, q) \geq \frac{1}{12} \text{KL}(p, q) / \log \max\{\frac{1}{q}, \frac{1}{1-q}\}$ .*

At this point we prove a version of Lemma 16 for a parameterized restricted case. We then reduce the general case to this lemma using an appropriate parameter. We note here that if we were to use the following parameterized lemma directly instead of Lemma 16, we would have obtained a direct generalization of the binary setting part of [5, Theorem 4.1].

► **Lemma 25.** *Let  $0 < q < \frac{1}{2}$ . Assume that for every  $1 \leq i \leq n$  and for every condition  $w \in \{0, 1\}^{i-1}$ , the prefix marginal  $\mu|_i^{x_{[i-1]}=w}$  is equivalent to  $\tau|_i^{x_{[i-1]}=w}$  or equivalent to  $\text{Ber}(p_{i,w})$  for some  $q \leq p_{i,w} \leq 1 - q$  (or both). Then  $\Delta_{\chi^2}(\tau, \mu) \geq \frac{1}{6} (d_{\text{TV}}(\tau, \mu))^2 / \log q^{-1}$ .*

**Proof.** By Pinsker's inequality and Lemma 23 we obtain:

$$2(d_{\text{TV}}(\tau, \mu))^2 \leq D_{\text{KL}}(\tau, \mu) = \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\text{KL}(\tau|_i^{x_{[i-1]}=w^{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w^{[i-1]}}(1))]$$

By our assumption, for every  $1 \leq i \leq n$ , if  $D_{\text{KL}}(\tau|_i^{x_{[i-1]}=w^{[i-1]}}, \mu|_i^{x_{[i-1]}=w^{[i-1]}}) \neq 0$  then the probability of  $\mu|_i^{x_{[i-1]}=w^{[i-1]}}$  to draw 1 is between  $q$  and  $1 - q$ . By Lemma 24 we obtain:

$$2(d_{\text{TV}}(\tau, \mu))^2 \leq \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\chi^2(\tau|_i^{x_{[i-1]}=w^{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w^{[i-1]}}(1)) \cdot 12 \log q^{-1}]$$

That is,

$$\frac{(d_{\text{TV}}(\tau, \mu))^2}{6 \log q^{-1}} \leq \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\chi^2(\tau|_i^{x_{[i-1]}=w^{[i-1]}}(1), \mu|_i^{x_{[i-1]}=w^{[i-1]}}(1))] = \Delta_{\chi^2}(\tau, \mu) \quad \blacktriangleleft$$

We could bound the KL-divergence of single bits using the  $\chi^2$  distance only because we assumed that the marginal probabilities of  $\mu$  are not too close to 0 or 1 (unless they are equal to their counterparts in  $\tau$ ). In the general case we cannot assume it, hence we need to instead consider a distribution  $\mu'$  which is close to  $\mu$  while satisfying this assumption.

We consider  $\mu$  as a “probability tree”, where the root represents the empty string, every edge represents an additional bit, and every leaf represents a complete sample. This tree (and hence the distribution  $\mu$ ) is fully determined using probabilities of the form  $\Pr_{x \sim \mu}[x_i = 1 | x_{[i-1]} = w]$ , where  $1 \leq i \leq n$  and  $w \in \{0, 1\}^{i-1}$ .

We construct another distribution  $\mu'$  based on such a tree pattern. For every  $1 \leq i \leq n$  and  $w \in \{0, 1\}^{i-1}$ , we set  $\Pr_{x \sim \mu'}[x_i = 1 | x_{[i-1]} = w]$  as follows:

$$\begin{aligned} & \min \left\{ \frac{d_{\text{TV}}(\tau, \mu)}{2n}, \Pr_{x \sim \tau}[x_i = 1 | x_{[i-1]} = w] \right\} & \text{if } \Pr_{x \sim \mu}[x_i = 1 | x_{[i-1]} = w] < \frac{d_{\text{TV}}(\tau, \mu)}{2n} \\ 1 - & \min \left\{ \frac{d_{\text{TV}}(\tau, \mu)}{2n}, \Pr_{x \sim \tau}[x_i = 0 | x_{[i-1]} = w] \right\} & \text{if } \Pr_{x \sim \mu}[x_i = 1 | x_{[i-1]} = w] > 1 - \frac{d_{\text{TV}}(\tau, \mu)}{2n} \\ & \Pr_{x \sim \mu}[x_i = 1 | x_{[i-1]} = w] & \text{otherwise} \end{aligned}$$

Observe that for every  $1 \leq i \leq n$  and  $w \in \{0, 1\}^{i-1}$ ,  $d_{\text{TV}}(\mu|_i^{x_{[i-1]}=w}, \mu'|_i^{x_{[i-1]}=w}) \leq \frac{d_{\text{TV}}(\tau, \mu)}{2n}$ . Hence,  $d_{\text{TV}}(\mu, \mu') \leq \Delta_{\text{TV}}(\mu, \mu') \leq \frac{1}{2}d_{\text{TV}}(\tau, \mu)$ . By the triangle inequality,  $d_{\text{TV}}(\tau, \mu') \geq \frac{1}{2}d_{\text{TV}}(\tau, \mu)$ .

Since the assumptions of Lemma 25 hold for  $\mu'$  (with  $q = \frac{1}{2n}d_{\text{TV}}(\tau, \mu)$ ), we can now conclude the proof of Lemma 16:

$$\begin{aligned} \Delta_{\chi^2}(\tau, \mu) &= \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\chi^2(\tau|_i^{x_{[i-1]}=w(i)}, \mu|_i^{x_{[i-1]}=w(i)})] \\ [\text{Monotonicity of } \chi^2] &\geq \sum_{i=1}^n \mathbb{E}_{w \sim \tau} [\chi^2(\tau|_i^{x_{[i-1]}=w(i)}, \mu'|_i^{x_{[i-1]}=w(i)})] \\ [\text{Lemma 25 with } q = \frac{1}{2n}d_{\text{TV}}(\tau, \mu)] &\geq (d_{\text{TV}}(\tau, \mu'))^2 / 6 \log \frac{2n}{d_{\text{TV}}(\tau, \mu)} \\ [d_{\text{TV}}(\tau, \mu') \geq \frac{1}{2}d_{\text{TV}}(\tau, \mu)] &\geq (d_{\text{TV}}(\tau, \mu))^2 / 24 \log \frac{2n}{d_{\text{TV}}(\tau, \mu)} \quad \blacktriangleleft \end{aligned}$$

## 6 Lower bound for Product

This section is devoted to the following lower bound:

► **Theorem 26.** *Every  $\varepsilon$ -test for PRODUCT must make at least  $\Omega(\sqrt{n}/\varepsilon^2)$  subcube queries.*

Let  $\pi_n$  denote the uniform distribution over  $\{0, 1\}^n$ . We show that distinguishing between  $\pi_n$  (which is in particular a product distribution) and a distribution that is  $\varepsilon$ -far from every product distribution requires  $\tilde{\Omega}(\sqrt{n}/\varepsilon^2)$  many queries.

Before we present our construction, we cite the corresponding lower bound for uniformity of a product distribution [8]:

► **Lemma 27** ([8]). *Let  $\mathcal{N}$  be the following distribution over inputs: draw  $b_1, \dots, b_n \sim \{+1, -1\}$  uniformly and independently, and return the distribution  $\prod_{i=1}^n \text{Ber}\left(\frac{1}{2} + b_i \frac{\varepsilon}{\sqrt{n}}\right)$ . Then the drawn input is always  $\Omega(\varepsilon)$ -far from  $\pi_n$ , and any unconditional sampling algorithm that distinguishes between inputs drawn from  $\mathcal{N}$  and  $\pi_n$  must take at least  $\Omega(\sqrt{n}/\varepsilon^2)$  many samples.*

In our construction, instead of adding a random bias for each coordinate, we partition the coordinates into pairs, and in each pair introduce a random “anti-product bias” as follows.

For  $b \in \{0, +1, -1\}$ , let  $\nu_b$  be the following distribution over  $\{0, 1\}^2$ :

$$\begin{aligned} \nu_b(00) &= \frac{1}{4} + b_i \frac{\varepsilon}{\sqrt{n}} & \nu_b(01) &= \frac{1}{4} - b_i \frac{\varepsilon}{\sqrt{n}} \\ \nu_b(10) &= \frac{1}{4} - b_i \frac{\varepsilon}{\sqrt{n}} & \nu_b(11) &= \frac{1}{4} + b_i \frac{\varepsilon}{\sqrt{n}} \end{aligned}$$

That is,  $\nu_0$  is the uniform distribution over two bits, and  $\nu_{+1}, \nu_{-1}$  are non-product distributions over two bits.

## 48:10 Improved Bounds for High-Dim. Testing Using Subcube Queries

Let  $\mathcal{Y}$  be the distribution over inputs that always returns the uniform distribution over  $\{0, 1\}^n$ , and let  $\mathcal{N}$  be the following distribution over “bad” inputs over  $\{0, 1\}^n$ : we partition  $[n]$  into fixed pairs (for example,  $(2i - 1, 2i)$  for every  $1 \leq i \leq \lfloor n/2 \rfloor$ ). For every pair we draw  $b_i \in \{+1, -1\}$  uniformly and independently. The distribution of the  $i$ th pair is  $\nu_{b_i}$ . All pairs are independent of each other. We assume that  $n$  is even, since for an odd  $n$  we can just assume that the  $n$ th bit is distributed uniformly and independently of the rest.

The following lemma, whose proof appears in Section 10, states that  $\mathcal{N}$  always draws a distribution far from PRODUCT.

► **Lemma 28.** *Every input distribution drawn from  $\mathcal{N}$  is  $\Omega(\varepsilon)$ -far from any product distribution.*

We now prove the indistinguishability of  $\mathcal{N}$  from the uniform distribution, starting with the sampling model, to which we later show a reduction from the subcube conditional model.

► **Lemma 29.** *An unconditional sampling algorithm that distinguishes between the uniform distribution and inputs that are drawn from  $\mathcal{N}$  must make  $\Omega(\sqrt{n}/\varepsilon^2)$  many queries.*

**Proof.** Without loss of generality, assume that  $n$  is even. Consider the following bijection over  $\{0, 1\}^2$ :  $(x, y) \rightarrow (x \oplus y, y)$ . For  $b \in \{+1, 0, -1\}$ , Let  $\nu'_b$  be the distribution that draws  $(x, y) \sim \nu_b$  and returns  $(x \oplus y, y)$ . Observe that  $\nu'_b$  is identical to the product distribution  $\text{Ber}(\frac{1}{2} + \frac{2b\varepsilon}{\sqrt{n}}) \times \text{Ber}(\frac{1}{2})$ . Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a bijection that applies the above mapping for every pair individually.

Let  $\mathcal{N}'$  be the distribution over inputs that draws  $\mu \sim \mathcal{N}$ , and then returns the distribution that draws  $x \sim \mu$  and returns  $f(x)$ . Note also that the distribution that drawn  $x \sim \pi_n$  and returns  $f(x)$  is identical to  $\pi_n$ . Since  $f$  is a bijection, this means that the number of samples needed to distinguish between the uniform distribution and  $\mathcal{N}$  is exactly the same as the number of samples needed to distinguish between the uniform distribution and  $\mathcal{N}'$ .

Note that the form of a distribution resulting from  $\mathcal{N}'$  is  $\prod_{i=1}^{n/2} \left( \text{Ber} \left( \frac{1}{2} + \frac{2b_i\varepsilon}{\sqrt{n}} \right) \times \text{Ber} \left( \frac{1}{2} \right) \right)$ . The restriction of  $\mathcal{N}'$  to odd indexes is identical to the construction of Lemma 27 (with parameters  $\sqrt{2}\varepsilon$  and  $n/2$ ), which requires at least  $\Omega(\sqrt{n}/\varepsilon^2)$  many unconditional samples to distinguish between it and the uniform distribution. Hence, this is also a lower bound for the number of samples needed to distinguish between the uniform distribution and inputs drawn from  $\mathcal{N}$ . ◀

► **Lemma 30.** *Let  $b \in \{0, +1, -1\}$ . Then for every subcube restriction  $q \in \{0, 1, *\}^2$ , we can simulate a  $q$ -conditioned sample from  $\nu_b$  by drawing a single unconditional sample from  $\nu_b$ , without having any knowledge about  $b$ .*

**Proof.** Let  $q \in \{0, +1, *\}^2$  and let  $b \in \{0, +1, -1\}$ . We first draw an unconditional sample  $x \sim \nu_b$ . There are three kinds of subcube restrictions:

- Trivial:  $q \in \{00, 01, 10, 11\}$ . To simulate such a query, we simply return  $q$ , ignoring the unconditional sample we have.
- Unconditional:  $q = **$ . To simulate such a query, we simply return  $x$ .
- Single restriction:  $q \in \{0*, 1*, *0, *1\}$ . We denote the parity of  $x$  by  $p = x|_1 \oplus x|_2$ , and construct the output as the concatenation of the two bits  $\text{out}_1$  and  $\text{out}_2$ , based on the following table:

|                |     |              |     |              |
|----------------|-----|--------------|-----|--------------|
| $q$            | 0*  | 1*           | *0  | *1           |
| $\text{out}_1$ | 0   | 1            | $p$ | $1 \oplus p$ |
| $\text{out}_2$ | $p$ | $1 \oplus p$ | 0   | 1            |

The correctness of the output in the trivial case and the unconditional case is trivial. We prove the correctness of the single-restriction case only for  $q = 0*$ , since the other cases are analogous.

$$\Pr_{\nu_b}[00|0*] = \frac{\Pr_{\nu_b}[00]}{\Pr_{\nu_b}[0*]} = \frac{\frac{1}{4} + \frac{b_i \varepsilon}{\sqrt{n}}}{\frac{1}{4} + \frac{b_i \varepsilon}{\sqrt{n}} + \frac{1}{4} - \frac{b_i \varepsilon}{\sqrt{n}}} = \frac{1}{2} + 2 \frac{b_i \varepsilon}{\sqrt{n}} = \Pr_{\nu_b}[00 \vee 11] = \Pr_{x \sim \nu_b}[x|_1 \oplus x|_2 = 0] \blacktriangleleft$$

At this point we can prove Theorem 26.

**Proof of Theorem 26.** By Lemma 28,  $\mathcal{N}$  draws an input distribution that is  $\Omega(\varepsilon)$ -far from any product distribution. Observe that since the structure of the pairs is known in advance and since they are independent, we can simulate every  $q$ -subcube-query algorithm using a  $q$ -unconditional-sample algorithm: for each query we draw a single sample and then use the simulation procedure of Lemma 30 for every pair in itself.

Since an unconditional test requires  $\Omega(\sqrt{n}/\varepsilon^2)$  queries to distinguish inputs drawn from  $\mathcal{N}$  from the uniform distribution by Lemma 29, and every subcube query to the uniform distribution or an input drawn from  $\mathcal{N}$  can be simulated using a single unconditional query, the lower bound holds for subcube algorithms as well.  $\blacktriangleleft$

## 7 Technical proofs for Equivalence testing

Here we prove the lemmas deferred from Section 5. We start with some helper lemmas.

► **Lemma 31.** *Let  $0 \leq p \leq q \leq 1$  for which  $p + q \leq 1$  and let  $X$  be the sum of  $N$  independent bits drawn from  $\text{Ber}(p)$ . Then,*

$$\Pr[X \geq \frac{1}{2}(p + q)N] \leq e^{-\frac{1}{12}\chi^2(p,q)}$$

**Proof.** Let  $\delta = q - p$ . If  $\delta \leq 2p$  then by Chernoff's bound:

$$\begin{aligned} \Pr[X \geq \frac{1}{2}(p + q)N] &= \Pr[X \geq (1 + \frac{\delta}{2p})\mathbb{E}[X]] \\ &\leq e^{-\frac{\delta^2}{12p^2}pN} = e^{-\frac{\delta^2}{12p}N} \leq e^{-\frac{(p-q)^2}{12(p+q)(2-(p+q))}N} = e^{-\frac{1}{12}\chi^2(p,q)N} \end{aligned}$$

If  $\delta > 2p$  then:

$$\begin{aligned} \Pr[X \geq \frac{1}{2}(p + q)N] &= \Pr[X \geq (1 + \frac{\delta}{2p})\mathbb{E}[X]] \\ &\leq e^{-\frac{\delta}{6p}pN} = e^{-\frac{\delta}{6}N} \leq e^{-\frac{(q-p)^2}{6(p+q)}} \leq e^{-\frac{1}{12}\chi^2(p,q)N} \blacktriangleleft \end{aligned}$$

► **Lemma 32.** *Let  $0 \leq p \leq q \leq 1$  for which  $p + q \leq 1$  and let  $X$  be the sum of  $N$  independent bits drawn from  $\text{Ber}(q)$ . Then,*

$$\Pr[X \leq \frac{1}{2}(p + q)N] \leq e^{-\frac{1}{8}\chi^2(p,q)} \leq e^{-\frac{1}{12}\chi^2(p,q)}$$

**Proof.** Let  $\delta = q - p \leq q$ . By Chernoff's bound:

$$\begin{aligned} \Pr[X \leq \frac{1}{2}(p + q)N] &= \Pr[X \leq (1 - \frac{\delta}{2q})\mathbb{E}[X]] \\ &\leq e^{-\frac{\delta^2}{8q^2}qN} = e^{-\frac{\delta^2}{8q}N} \leq e^{-\frac{(p-q)^2}{8(p+q)(2-(p+q))}N} = e^{-\frac{1}{8}\chi^2(p,q)N} \blacktriangleleft \end{aligned}$$

## 48:12 Improved Bounds for High-Dim. Testing Using Subcube Queries

► **Lemma 33.** *Let  $p, q \in [0, 1]$ . Let  $X$  be the sum of  $N$  independent bits drawn from  $\text{Ber}(p)$  and  $Y$  be the sum of  $N$  independent bits drawn from  $\text{Ber}(q)$ . Then, with probability at least  $\left(1 - e^{-\frac{1}{12}\chi^2(p,q)N}\right)^2$ , the sign of  $X - Y$  matches the sign of  $p - q$ .*

**Proof.** Without loss of generality,  $p + q \leq 1$  (otherwise use  $1 - q$  and  $1 - p$  instead of  $p$  and  $q$ , noting that  $d(1 - q, 1 - p) = d(p, q)$  and that the sign of  $(1 - q) - (1 - p)$  matches the sign of  $p - q$ ).

If  $p = q$  then the lemma is vacuously correct. If  $p < q$ , then by Lemma 31,

$$\begin{aligned} \Pr[X - Y < 0] = \Pr[X < Y] &\geq \Pr[X < \frac{1}{2}(p + q)N] \Pr[Y > \frac{1}{2}(p + q)N] \\ &\geq \left(1 - e^{-\frac{1}{12}\chi^2(p,q)N}\right)^2 \end{aligned}$$

The case where  $p > q$  is analogous, using Lemma 32. ◀

We recall and prove Lemma 17:

► **Lemma 17.** *Let  $p, q \in [0, 1]$  be two probabilities. Given unconditional sampling access to  $\text{Ber}(p)$  and  $\text{Ber}(q)$ , we can distinguish, with probability  $\frac{2}{3}$ , between the case where  $p = q$  and the case where  $\chi^2(p, q) > \varepsilon$ , at the cost of  $O(1/\varepsilon)$  samples from each of them.*

**Proof.** We repeat the following procedure 64 times: let  $N = \lceil 24/\varepsilon \rceil$ . Also, let  $X$  be the sum of  $N$  independent samples drawn from  $\text{Ber}(p)$  and  $Y$  be the sum of  $N$  independent samples drawn from  $\text{Ber}(q)$ . To conclude a single trial we check whether  $A > B$  or  $A < B$  (or neither) holds.

Let  $A$  be the number of trials with  $X > Y$  and  $B$  be the number of trials with  $X < Y$ . If  $|A|, |B| \leq 40$  we accept ( $p = q$ ), and otherwise we reject ( $\chi^2(p, q) > \varepsilon$ ).

If  $p = q$ , then by symmetry,  $\mathbb{E}[X < Y] = \mathbb{E}[X > Y] \leq \frac{1}{2}$ . That is,  $\mathbb{E}[A] = \mathbb{E}[B] \leq 32$ . By Chernoff's bound,  $\Pr[A \geq 41] < e^{-2 \cdot 9^2/64} < \frac{1}{6}$  and  $\Pr[B \geq 41] < \frac{1}{6}$ . Hence, the probability to accept is at least  $1 - \frac{2}{6} = \frac{2}{3}$ .

If  $\chi^2(p, q) > \varepsilon$ , then by Lemma 33, one of  $\Pr[X < Y]$  and  $\Pr[X > Y]$  is at least  $(1 - e^{-\frac{1}{12}\chi^2(p,q)N})^2 \geq (1 - e^{-2})^2 > 0.74$ . Without loss of generality, we assume that  $p < q$ . In this case,  $\mathbb{E}[A] > 47.36$ . By the Chernoff bound, the probability to reject is at least  $1 - \Pr[A \leq 40] \geq 1 - e^{-2 \cdot 7.36^2/64} > \frac{2}{3}$ . ◀

► **Lemma 34** (Well known). *For  $p, q \in [0, 1]$ ,  $\text{KL}(p, q) \leq \frac{(p-q)^2}{q(1-q)}$ . More formally, in  $(0, 1) \times (0, 1)$ , the ratio between these expressions is a non-negative continuous function that is bounded by 1.*

► **Lemma 35** (Direct corollary). *Let  $p, q \in [0, 1]$ . If  $p = aq$  for some real  $a$ , then*

$$\text{KL}(aq, q) \leq \frac{(a-1)^2}{1-q}q$$

Finally, we recall and prove Lemma 24:

► **Lemma 24.** *Let  $p, q \in [0, 1]$ . Then  $\chi^2(p, q) \geq \frac{1}{12} \text{KL}(p, q) / \log \max\{\frac{1}{q}, \frac{1}{1-q}\}$ .*

**Proof.** We actually prove that  $\chi^2(p, q) / \text{KL}(p, q) \geq \frac{1}{12 \log \max\{\frac{1}{q}, \frac{1}{1-q}\}}$  for every  $p \in [0, 1]$  and  $q \in (0, 1)$ . For the edge cases of  $q$  (which we do not use in our proofs anyway except when  $p = q$ ), the bound remains correct by considering the limit of  $\text{KL}(p, q) / \log \max\{\frac{1}{q}, \frac{1}{1-q}\}$ .

Without loss of generality,  $q \leq \frac{1}{2}$ . We can assume so since  $\chi^2(p, q) = \chi^2(1-p, 1-q)$ ,  $\text{KL}(p, q) = \text{KL}(1-p, 1-q)$  and  $\max\{\log 1/q, \log 1/(1-q)\} = \max\{\log 1/(1-q), \log 1/(1-(1-q))\}$ . Based on this assumption it is sufficient to show that  $\frac{\chi^2(p, q)}{\text{KL}(p, q)} \geq \frac{1}{12 \log q^{-1}}$ .

Let  $a = p/q$  ( $0 < a \leq 1/q$ ). If  $a \geq 2$ :

$$\begin{aligned}\chi^2(aq, q) &\geq \frac{(a-1)^2}{2(a+1)q} \\ \text{KL}(aq, q) &\leq p \log \frac{p}{q} \leq a \log q^{-1} \cdot q \\ \frac{\chi^2(aq, q)}{\text{KL}(aq, q)} &\geq \frac{(a-1)^2}{2a(a+1) \log q^{-1}} \geq \frac{1}{12 \log q^{-1}}\end{aligned}$$

If  $a \leq 2$ :

$$\begin{aligned}\chi^2(aq, q) &\geq \frac{(a-1)^2}{2(a+1)q} \\ \text{KL}(aq, q) &\leq \frac{(a-1)^2}{1-q}q \\ \frac{\chi^2(aq, q)}{\text{KL}(aq, q)} &\geq \frac{1-q}{2(a+1)} \geq \frac{1}{12 \log q^{-1}}\end{aligned}$$

where the very last inequality uses the assumption that  $q \leq \frac{1}{2}$ . ◀

## 8 Extending the Equivalence test to general alphabets

We extend the definitions of the prefix oracle to non-binary settings.

► **Definition 36** (Subcube oracle access in non-binary strings). *Let  $\mu$  be an unknown distribution over  $\prod_{i=1}^n \Omega_n$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. The subcube oracle has as input a tuple  $(A_1, \dots, A_n)$  where  $A_i \subseteq \Omega_i$  for every  $1 \leq i \leq n$ . The output distributes as  $\mu|_{\prod_{i=1}^n A_i}$ . For technical reasons, if  $\Pr_\mu[\prod_{i=1}^n A_i] = 0$ , then the oracle indicates an error. Note that the answers of the oracle are fully independent of the answers that were given to previous queries.*

In the corresponding definition for a prefix oracle, we still demand that until the “break-off index”  $i$  all conditions force single outcomes from the sets  $\Omega_j$ , while after the break-off index there are no restrictions at all. However, at index  $i$  we allow conditions to any subset of  $\Omega_i$  to take place. There is no such distinction in the binary case, where  $|\Omega_i| = 2$  and hence all non-trivial conditions are to a single outcome.

► **Definition 37** (Prefix oracle access in non-binary settings). *Let  $\mu$  be an unknown distribution over  $\prod_{i=1}^n \Omega_n$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. The input of the prefix oracle consists of an index  $1 \leq i \leq n$ , which we refer to as the index of the prefix, elements  $a_j \in \Omega_j$  for every  $1 \leq j \leq i-1$ , and a condition  $A \subseteq \Omega_i$ . The output of the oracle distributes like  $\mu|_{\{x: x_i \in A \wedge x_1 = a_1, \dots, x_{i-1} = a_{i-1}\}}$ .*

► **Definition 38** (Marginal subcube oracle access in non-binary settings). *Let  $\mu$  be an unknown distribution over  $\prod_{i=1}^n \Omega_n$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. The marginal subcube oracle has as input an index  $i$  and a tuple  $(A_1, \dots, A_n)$  where  $A_j \subseteq \Omega_j$  for every  $1 \leq j \leq n$ . The output distributes as  $\mu|_{\prod_{j=1}^n A_j}$ . For technical reasons, if  $\Pr_\mu[\prod_{i=1}^n A_i] = 0$ , then the oracle indicates an error.*

► **Definition 39** (Marginal prefix oracle access in non-binary settings). Let  $\mu$  be an unknown distribution over  $\prod_{i=1}^n \Omega_i$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. The input of the marginal prefix oracle consists of an index  $1 \leq i \leq n$ , which we refer to as the index of the prefix, elements  $a_j \in \Omega_j$  for every  $1 \leq j \leq i-1$ , and a condition  $A \subseteq \Omega_i$ . The output of the oracle distributes like  $\mu|_{\{x: x_i \in A \wedge x_1 = a_1, \dots, x_{i-1} = a_{i-1}\}}$ .

Based on these definitions, we state Theorem 40:

► **Theorem 40.** Let  $\mu$  and  $\tau$  be two distributions over  $\prod_{i=1}^n \Omega_i$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. If  $\mu$  is accessible through the marginal prefix oracle and  $\tau$  is accessible through the prefix oracle, then we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using  $\tilde{O}(\sum_{i=1}^n \log_2 |\Omega_i| / \varepsilon^2)$  queries.

Before we prove Theorem 40, we need the following.

► **Lemma 41** (Binary form of a composite distribution). Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ , where  $\Omega_1, \dots, \Omega_n$  are non-empty finite sets. There exists a distribution  $\mu^*$  over the set  $\{0, 1\}^{\sum_{i=1}^n \lceil \log_2 |\Omega_i| \rceil}$  that is equivalent to  $\mu$  up to relabeling, for which every subcube (respectively prefix) query to  $\mu^*$  can be simulated using a single subcube (respectively prefix) query to  $\mu$ , and every marginal subcube (respectively prefix) query to  $\mu^*$  can be simulated using a single marginal subcube (respectively prefix) query to  $\mu$ .

**Proof.** For every  $1 \leq i \leq n$ , let  $f_i : \Omega_i \rightarrow \{0, 1\}^{\lceil \log_2 |\Omega_i| \rceil}$  be an arbitrary injective function from  $\Omega_i$  to  $\{0, 1\}^{\lceil \log_2 |\Omega_i| \rceil}$ . Let  $f : \prod_{i=1}^n \Omega_i \rightarrow \{0, 1\}^{\sum_{i=1}^n \lceil \log_2 |\Omega_i| \rceil}$  be the concatenation of these mappings. More precisely,  $f((x_1, \dots, x_n)) = \text{concatenate}(f_1(x_1), \dots, f_n(x_n))$ . Let  $N = \sum_{i=1}^n \lceil \log_2 |\Omega_i| \rceil$ .

For every  $1 \leq i \leq n$ , we define a projection function  $g_i : \{0, 1\}^{\sum_{j=1}^n \lceil \log_2 |\Omega_j| \rceil} \rightarrow \Omega_i$  by

$$g_i(x) = (f_i)^{-1} \left( x|_{\{\sum_{j=1}^{i-1} \lceil \log_2 |\Omega_j| \rceil + 1, \dots, \sum_{j=1}^i \lceil \log_2 |\Omega_j| \rceil\}} \right)$$

where  $g_i(x)$  is defined arbitrarily if the corresponding binary string is not in  $f_i$ 's image, as this will be a zero-probability event. Observe that for every  $(x_1, \dots, x_n) \in \prod_{i=1}^n \Omega_i$  and for every  $1 \leq i \leq n$ ,  $x_i = g_i(f(x_1, \dots, x_n))$ .

Let  $\mu^*$  be the distribution over  $\{0, 1\}^{\sum_{i=1}^n \lceil \log_2 |\Omega_i| \rceil}$  that draws  $x \sim \mu$  and returns  $f(x)$ . Since  $f$  is an injective function,  $\mu^*$  is equivalent to  $\mu$  up to relabeling.

For simulating subcube queries consider some  $I^* \subseteq [N]$  and  $w^* \in \{0, 1\}^{I^*}$ . For every  $1 \leq i \leq n$  and  $x^* \in \{0, 1\}^N$ , let  $x_i = g_i(x^*)$  (that is, we decompose  $x^*$  using  $x^* = f(x_1, \dots, x_n)$ ). Also, for every  $1 \leq i \leq n$ , let  $A_i(I^*, w^*) = \{g_i(s^*) : s^* \in \{0, 1\}^N \wedge s^*|_{I^*} = w^*\}$ . Based on this composition, we obtain:

$$\{x^* \in \{0, 1\}^N : x^*|_{I^*} = w^*\} = \left\{ f(x_1, \dots, x_n) : \bigwedge_{i=1}^n (x_i \in A_i(I^*, w^*)) \right\}$$

The last expression is the  $f$ -image of all elements in the  $\mu$ -subcube condition  $\prod_{i=1}^n A_i(I^*, w^*)$ . Hence, every subcube query of  $\mu^*$  can be simulated using a single subcube query to  $\mu$ .

Observe that this construction preserves prefix queries. That is, if a subcube query to  $\mu^*$  is a prefix query, then the simulated subcube query to  $\mu$  is a prefix query as well. Note that this argument holds for marginal queries as well, since we can extract the relevant bit of the sampled coordinate. ◀

Theorem 40 now follows.



**Proof of Theorem 40.** Let  $\mu$  and  $\tau$  be two distributions over  $\prod_{i=1}^n \Omega_i$ , where  $\Omega_1, \dots, \Omega_n$  are all finite. We use Lemma 41 to define two distributions  $\mu^*, \tau^*$  that are identical to  $\mu, \tau$  respectively up to relabeling (which is the same in both constructions). Based on this lemma:

- $d_{\text{TV}}(\tau^*, \mu^*) = d_{\text{TV}}(\tau, \mu)$ , since  $\mu^*, \tau^*$  are the same as  $\mu, \tau$  up to relabeling (which is the same for both constructions  $\mu \rightarrow \mu^*$  and  $\tau \rightarrow \tau^*$ ).
- Every prefix query to  $\tau^*$  can be simulated using a single prefix query to  $\tau$ .
- Every marginal prefix query to  $\mu^*$  can be simulated using a single marginal prefix query to  $\mu$ .

Hence, we can distinguish between  $\tau = \mu$  and  $d_{\text{TV}}(\tau, \mu) > \varepsilon$  using Theorem 20 with the input  $(\mu^*, \tau^*, \varepsilon)$  by simulating every query to  $\tau^*$  or  $\mu^*$  through a single query to the corresponding input distribution  $\tau$  or  $\mu$ . ◀

## 9 Upper bound for Product

We reduce a test for PRODUCT to a test for EQUIVALENCE, based on the following key observation:

► **Observation 42.** *Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ . If  $\mu$  is  $\varepsilon$ -far from any product distribution, then in particular it is  $\varepsilon$ -far from the product of its marginals,  $\prod_{i=1}^n \mu|_i$ .*

In the binary setting, simulating a marginal prefix query to the product of marginals is pretty straightforward and can be done using one unconditional query to  $\mu$ , which is in particular a prefix query:

► **Observation 43.** *Let  $\mu$  be a distribution over  $\{0,1\}^n$ , and let  $\mu' = \prod_{i=1}^n \mu|_i$  be the product of  $\mu$ 's marginals. Then we can simulate every marginal prefix query to  $\mu'$  using one unconditional sample from  $\mu$  (and keeping its  $i$ th entry).*

In the general setting, we need the stronger subcube access. The reason is that when taking a prefix marginal query at index  $i$  from the binary representation of  $\mu' = \prod_{j=1}^n \mu|_j$ , it may be that that  $i$  is “in the middle” of the  $\lceil \log |\Omega_j| \rceil$ -bit representation of  $j$ th coordinate of  $\mu'$ , and then this query must translate to a non-trivial set  $A_j \subseteq \Omega_j$  when simulating it using query access to  $\mu$ .

► **Observation 44.** *Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ , and let  $\mu' = \prod_{i=1}^n \mu|_i$  be the product of  $\mu$ 's marginals. Then we can simulate every marginal prefix query to  $\mu'$  using one subcube query to  $\mu$ .*

At this point, we can prove Theorem 45.

► **Theorem 45.** *Let  $\mu$  be a distribution over  $\prod_{i=1}^n \Omega_i$ . For every  $0 < \varepsilon < 1$ , we can distinguish between the case where  $\mu$  is a product distribution and the case where it is  $\varepsilon$ -far from every product distribution at the cost of  $\tilde{O}(\sum_{i=1}^n \log |\Omega_i| / \varepsilon^2)$  subcube queries. Moreover, if  $|\Omega_i| = 2$  for every  $1 \leq i \leq n$ , then all these queries are prefix queries.*

**Proof.** Let  $\mu' = \prod_{i=1}^n \mu|_i$  be the product of  $\mu$ 's marginals. If  $\mu$  is a product distribution then  $\mu' = \mu$ , and if  $\mu$  is  $\varepsilon$ -far from every product distribution, then in particular  $d_{\text{TV}}(\mu, \mu') > \varepsilon$ .

Since  $\mu$  is accessible through the subcube oracle, we can use Observation 44 to simulate every marginal prefix query to  $\mu'$  at the cost of one subcube query to  $\mu$ . If  $|\Omega_i| = 2$  for every  $1 \leq i \leq n$ , then we can use an unconditional sample instead of a subcube query by Observation 43.

Hence, we can reduce the  $\varepsilon$ -test of  $\mu$  for PRODUCT to an  $\varepsilon$ -test of the equivalence of  $\mu'$  and  $\mu$ , which we perform using Theorem 40. As noted above, this produces subcube queries for general alphabets, and only prefix queries for the binary setting. ◀

## 10 Technical proofs for the Product lower bound

Recall that we denote the uniform distribution over  $\{0, 1\}^n$  by  $\pi_n$ .

Before we prove Lemma 28, we need the following technical lemmas.

► **Lemma 46.** *For a string  $y \in \{0, 1\}^n$ , let  $C_1(y)$  be the number of 1s in  $y$ . If  $n \geq 70$ , then  $\Pr_{y \sim \pi_n} [C_1(y) > \frac{1}{2}n + \frac{1}{4}\sqrt{n}] \geq \frac{1}{4}$ .*

**Proof.** We use the well-known bound  $\binom{n}{\lfloor n/2 \rfloor} \leq 2^n \cdot \sqrt{\frac{2}{\pi n}}$  (for all  $n \geq 1$ ) to obtain for  $n \geq 70$ :

$$\begin{aligned} \Pr_{y \sim \pi_n} \left[ \left| C_1(y) - \frac{1}{2}n \right| \leq \frac{1}{4}\sqrt{n} \right] &= \sum_{k=\lceil \frac{1}{2}n - \frac{1}{4}\sqrt{n} \rceil}^{\lfloor \frac{1}{2}n + \frac{1}{4}\sqrt{n} \rfloor} \Pr_{y \sim \pi_n} [C_1(y) = k] \\ \text{[Since } \binom{n}{k} \leq \binom{n}{\lfloor n/2 \rfloor}] &\leq \left( 2 \cdot \frac{1}{4}\sqrt{n} + 1 \right) 2^{-n} \binom{n}{\lfloor n/2 \rfloor} \\ &\leq \left( \frac{1}{2}\sqrt{n} + 1 \right) \frac{\sqrt{2}}{\sqrt{\pi n}} \leq \frac{1}{2} \end{aligned}$$

By symmetry reasons, if  $n \geq 70$  then:

$$\begin{aligned} \Pr_{y \sim \pi_n} \left[ C_1(y) > \frac{1}{2}n + \frac{1}{4}\sqrt{n} \right] &= \frac{1}{2} \Pr_{y \sim \pi_n} \left[ \left| C_1(y) - \frac{1}{2}n \right| > \frac{1}{4}\sqrt{n} \right] \\ &= \frac{1}{2} \left( 1 - \Pr_{y \sim \pi_n} \left[ \left| C_1(y) - \frac{1}{2}n \right| \leq \frac{1}{4}\sqrt{n} \right] \right) \geq \frac{1}{4} \quad \blacktriangleleft \end{aligned}$$

► **Lemma 47.** *Let  $0 < \delta < \frac{1}{4\sqrt{n}}$  and  $\tau$  be a product distribution over  $\{0, 1\}^n$ . For sufficiently large  $n$ , if  $|\Pr_{\tau}[x_i = 1] - \frac{1}{2}| > \delta$  for every  $1 \leq i \leq n$ , then the distance of  $\tau$  from the uniform distribution over  $\{0, 1\}^n$  is at least  $\frac{1}{16}\delta\sqrt{n}$ .*

**Proof.** Without loss of generality, we can assume that  $\Pr_{\tau}[x_i = 1] \geq \frac{1}{2}$  for every  $1 \leq i \leq n$ . Otherwise, we can negate the “wrong” bits while preserving the distance from the uniform distribution. Based on this assumption, we have a product distribution whose probability to draw 1 at any individual index is at least  $\frac{1+\delta}{2}$ .

Let  $\tau'$  be the product distribution whose probability to draw 1 at any index is exactly  $\frac{1+\delta}{2}$ . Observe that the distance of  $\tau'$  from the uniform distribution is a lower bound of the distance of  $\tau$  from the uniform distribution.

Let  $y \in \{0, 1\}^n$  be a string, and let  $C_1(y)$  (respectively  $C_0(y)$ ) be the number of 1s (respectively 0s) in  $y$ . If  $C_1(y) \geq \frac{1}{2}n + \frac{1}{4}\sqrt{n}$ , then:

$$\begin{aligned} \frac{\Pr_{\tau'}[y]}{\Pr_{\pi_n}[y]} &= (1 + \delta)^{C_1(y)} \cdot (1 - \delta)^{n - C_1(y)} \\ &= (1 + \delta)^{C_1(y) - C_0(y)} \cdot ((1 + \delta)(1 - \delta))^{C_0(y)} \\ &\geq (1 + \delta)^{\sqrt{n}/2} \cdot ((1 + \delta)(1 - \delta))^{\frac{1}{2}n - \frac{1}{4}\sqrt{n}} \\ &= (1 + \delta)^{\sqrt{n}/2} \cdot (1 - \delta^2)^{\frac{1}{2}n - \frac{1}{4}\sqrt{n}} \\ &\geq (1 + \delta)^{\sqrt{n}/2} \cdot (1 - \delta^2)^{\frac{1}{2}n} \\ [(1 + a)^b \geq 1 + ab \text{ for } b \geq 1, |a| < 1] &\geq \left( 1 + \frac{1}{2}\delta\sqrt{n} \right) \cdot \left( 1 - \frac{1}{2}\delta^2 n \right) \\ [\delta < \frac{1}{4\sqrt{n}}] &\geq 1 + \frac{1}{4}\delta\sqrt{n} \end{aligned}$$

In particular,

$$\frac{\Pr_{y \sim \tau'} [C_1(y) > \frac{1}{2}n + \frac{1}{4}\sqrt{n}]}{\Pr_{y \sim \pi_n} [C_1(y) > \frac{1}{2}n + \frac{1}{4}\sqrt{n}]} \geq 1 + \frac{1}{4}\delta\sqrt{n}$$

Since  $\Pr_{y \sim \pi_n} [C_1(y) > \frac{1}{2}n + \frac{1}{4}\sqrt{n}] \geq \frac{1}{4}$  by Lemma 46, we obtain that  $d_{\text{TV}}(\tau', \pi_n) \geq \frac{1}{4}\delta\sqrt{n} \cdot \frac{1}{4} = \frac{1}{16}\delta\sqrt{n}$ .  $\blacktriangleleft$

We recall and prove Lemma 28:

**► Lemma 28.** *Every input distribution drawn from  $\mathcal{N}$  is  $\Omega(\varepsilon)$ -far from any product distribution.*

**Proof.** Without loss of generality, we assume that  $n$  is even. Let  $b_1, \dots, b_{n/2} \in \{+1, -1\}$  be the parameters of the construction, that is, the drawn input is  $\mu = \prod_{i=1}^{n/2} \nu_{b_i}$ . Recall that for  $b \in \{+1, -1\}$ ,  $\Pr_{\nu_b}[x_1 = 1] = \Pr_{\nu_b}[x_2 = 1] = \frac{1}{2}$  and  $\Pr_{\nu_b}[x_1 \oplus x_2 = 1] = \frac{1}{2} - \frac{2b\varepsilon}{\sqrt{n}}$ . We assume that  $n$  is sufficiently large so that  $\lfloor n/6 \rfloor$  satisfies the constraints of Lemma 47.

For a given product distribution  $\tau$ , for every  $1 \leq i \leq n/2$ , let:

$$\begin{aligned} \delta_{i,0} &= \Pr_{\tau}[x_{2i} = 1] - \frac{1}{2} \\ \delta_{i,1} &= \Pr_{\tau}[x_{2i-1} = 1] - \frac{1}{2} \\ \delta_{i,2} &= \Pr_{\tau}[x_{2i} \oplus x_{2i-1} = 1] - \frac{1}{2} \end{aligned}$$

Let  $I_0$  be the set of indexes for which  $|\delta_{i,0}| > \frac{\varepsilon}{10\sqrt{n}}$ . Let  $I_1$  be defined analogously for  $\delta_{i,1}$ , and let  $I_2 = [n/2] \setminus (I_0 \cup I_1)$  be the set of all other indexes. Since  $|I_0| + |I_1| + |I_2| = n/2$ , at least one of them contains at least  $\frac{1}{6}n$  elements.

**Case I.**  $|I_0| \geq \frac{1}{6}n$ . Let  $I = \{2i : i \in I_0\}$ . Observe that  $\mu|_I$  is uniform over  $\{0, 1\}^{n/2}$ , since  $\mu$  is the product of  $n/2$  independent distributions over pairs whose marginals are  $\frac{1}{2}$ , and every pair contributes exactly one index. According to Lemma 47,

$$d_{\text{TV}}(\tau, \mu) \geq d_{\text{TV}}(\tau|_I, \mu|_I) \geq \frac{1}{16} \cdot \frac{\varepsilon}{10\sqrt{n}} \cdot \sqrt{\frac{1}{6}n} = \frac{1}{160\sqrt{6}}\varepsilon > \frac{1}{400}\varepsilon$$

**Case II.**  $|I_1| \geq \frac{1}{6}n$ . Completely analogous to Case I. Again  $d_{\text{TV}}(\tau, \mu) > \frac{1}{400}\varepsilon$ .

**Case III.**  $|I_2| \geq \frac{1}{6}n$ . For every  $i \in I_2$ ,  $|\delta_{i,0}|, |\delta_{i,1}| \leq \frac{\varepsilon}{10\sqrt{n}}$ . Hence,

$$\begin{aligned} |\delta_{i,2}| &= \left| \Pr_{\tau}[x_{2i-1} \oplus x_{2i} = 1] - \frac{1}{2} \right| = \left| \left( \frac{1}{2} - \delta_{i,0} \right) \left( \frac{1}{2} + \delta_{i,1} \right) + \left( \frac{1}{2} + \delta_{i,0} \right) \left( \frac{1}{2} - \delta_{i,1} \right) - \frac{1}{2} \right| \\ &= 2|\delta_{i,0}||\delta_{i,1}| \leq \frac{\varepsilon^2}{50n} \end{aligned}$$

Let  $I = \{2i : i \in I_2\} \cup \{2i-1 : i \in I_2\}$ . Let  $f : \{0, 1\}^I \rightarrow \{0, 1\}^{|I_2|}$  be the function that maps every pair in  $I_2$  to its parity bit. In other words, for every  $i \in I_2$ , the bits  $x_{2i-1}, x_{2i}$  are mapped to a single bit  $x_{2i-1} \oplus x_{2i}$ . Let  $\mu'$  (respectively  $\tau'$ ) be the distribution over  $\{0, 1\}^{|I_2|}$  that draws a sample  $x \sim \mu$  (respectively  $x \sim \tau$ ) and returns  $f(x)$ . Observe that both  $\mu'$  and  $\tau'$  are product distributions over  $\{0, 1\}^{|I_2|}$ , since the pairs are independent and every pair is mapped into a single bit.

Note that  $d_{\text{TV}}(\tau', \pi_{|I_2|}) \leq \Delta_{\text{TV}}(\tau', \pi_{|I_2|}) = \sum_{i=1}^{|I_2|} d_{\text{TV}}(\tau'|_i, \text{Ber}(1/2)) \leq |I_2| \cdot \frac{\varepsilon^2}{50n} \leq \frac{1}{50}\varepsilon^2$ . In  $\mu'$ , by definition of the  $\nu_b$ s, all marginals have the form  $\frac{1}{2} \pm \frac{\varepsilon}{\sqrt{n}}$ , hence by Lemma 47,

$$d_{\text{TV}}(\mu', \pi_{|I_2|}) \geq \frac{1}{16} \cdot \frac{\varepsilon}{\sqrt{n}} \cdot \sqrt{\frac{1}{6}n} = \frac{\varepsilon}{16\sqrt{6}} \geq \frac{1}{40}\varepsilon$$

By a data processing inequality and the triangle inequality,

$$d_{\text{TV}}(\mu, \tau) \geq d_{\text{TV}}(\mu', \tau') \geq d_{\text{TV}}(\mu', \pi_{|I_2|}) - d_{\text{TV}}(\tau', \pi_{|I_2|}) \geq \frac{1}{40}\varepsilon - \frac{1}{50}\varepsilon^2 > \frac{1}{400}\varepsilon \quad \blacktriangleleft$$

---

## References

- 1 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001.
- 2 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE, 2000.
- 3 Arnab Bhattacharyya, Sutanu Gayen, Saravanan Kandasamy, and NV Vinodchandran. Testing product distributions: A closer look. In *Algorithmic Learning Theory*, pages 367–396. PMLR, 2021.
- 4 Rishiraj Bhattacharyya and Sourav Chakraborty. Property testing of joint distributions using conditional samples. *CoRR*, abs/1702.01454, 2017. [arXiv:1702.01454](https://arxiv.org/abs/1702.01454).
- 5 Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda. Complexity of high-dimensional identity testing with coordinate conditional sampling. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1774–1790. PMLR, 2023.
- 6 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.
- 7 Clément L Canonne, Xi Chen, Gautam Kamath, Amit Levi, and Erik Waingarten. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 321–336. SIAM, 2021.
- 8 Clément L Canonne, Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Testing bayesian networks. In *Conference on Learning Theory*, pages 370–448. PMLR, 2017.
- 9 Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015.
- 10 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 561–580, 2013.
- 11 Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. Learning and testing junta distributions with sub cube conditioning. In *Conference on Learning Theory*, pages 1060–1113. PMLR, 2021.
- 12 Xi Chen and Cassandra Marcussen. Uniformity testing over hypergrids with subcube conditioning. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4338–4370. SIAM, 2024.
- 13 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.
- 14 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75, 2011.
- 15 Oded Goldreich and Dana Ron. Testing distributions of huge objects. *TheoretCS*, 2, 2023.
- 16 Leonid A Levin. One-way functions and pseudorandom generators. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 363–365, 1985.

- 17 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.
- 18 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

## A Explicit algorithms

We provide here explicit representations for the binary setting algorithms constructed in this paper.

### Levin’s work-balance method

For a random variable  $X$  that is bounded between 0 and 1, it distinguishes between  $E[X] = 0$  and  $E[X] > \varepsilon$ . To do that, we use another random variable  $Y$ , for which we can distinguish between  $E[X|Y = y] = 0$  and  $E[X|Y = y] > \rho$  for every  $0 < \rho < 1$ , at a cost of  $O(1/\rho)$ . Overall, the cost of the algorithm is  $O(\log^2 \varepsilon^{-1}/\varepsilon)$ . The original construction is found in [16] and the following optimized version appears as an exercise in [13].

The black box is run a logarithmic number of times for every  $y$  that we draw (as opposed to once in, for example, [13, Exercise 8.4]) since it refers here to a procedure with two-sided error.

■ **Algorithm 1** Levin’s work-balance procedure.

---

**input**  $Y$  – a random variable, accessible through unconditional sampling.  
**input**  $\varepsilon$  – a threshold parameter.  
**input** A random black box that, for every  $y \in \text{supp}(Y)$  and  $0 < \varepsilon' < 1$ :  
    **completeness** if  $E[X|Y = y] = 0$ , it accepts with at least probability  $2/3$ .  
    **soundness** if  $E[X|Y = y] > \varepsilon'$ , it rejects with probability at least  $2/3$ .  
    **resource cost complexity**  $O(1/\varepsilon')$ .  
**completeness** If  $E[X] = 0$ , then the output is ACCEPT with probability at least  $2/3$ .  
**soundness** If  $E[X] > \varepsilon$ , then the output is REJECT with probability at least  $2/3$ .  
**resource cost complexity**  $O(\log^2 \varepsilon^{-1}/\varepsilon)$ .  
**for**  $t$  from 1 to  $\lceil \log_2(2/\varepsilon) \rceil$  **do**  
    **let**  $\varepsilon' \leftarrow 2^{-t}$ .  
    **for**  $\lceil 2^{3-t} \varepsilon^{-1} \rceil$  **times do**  
        **draw**  $y \sim Y$ .  
        **set**  $r \leftarrow 0$ .  
        **for**  $\lceil 64(\log_2 \varepsilon^{-1} + 2) \rceil$  **times do**  
            **run** the black box with  $(y, \varepsilon')$ .  
            **if** the black box accepts **then**  
                **set**  $r \leftarrow r + 1$ .  
            **else**  
                **set**  $r \leftarrow r - 1$ .  
        **if**  $r < 0$  **then**  
            **return** REJECT.  
**return** ACCEPT.

---

**$\chi^2$ -test of single-bit distributions**

The following is the algorithm that detects the difference between two Bernoulli distributions as it was described in Lemma 17.

■ **Algorithm 2**  $\chi^2$ -test of single-distributions.

---

**input** Two Bernoulli distributions  $\text{Ber}(p)$  and  $\text{Ber}(q)$ , accessible through samples.  
**completeness** If  $p = q$ , then the algorithm accepts with probability at least  $2/3$ .  
**soundness** If  $\chi^2(p, q) > \varepsilon$ , then the algorithm rejects with probability at least  $2/3$ .  
**complexity**  $O(1/\varepsilon)$  samples.  
**let**  $N \leftarrow \lceil 16/\varepsilon \rceil$ .  
**set**  $A, B \leftarrow 0$ .  
**for** 64 **times do**  
    **draw**  $N$  independent samples from  $\text{Ber}(p)$ , let  $X$  be their sum.  
    **draw**  $N$  independent samples from  $\text{Ber}(q)$ , let  $Y$  be their sum.  
    **if**  $X > Y$  **then**  
        **set**  $A \leftarrow A + 1$ .  
    **if**  $X < Y$  **then**  
        **set**  $B \leftarrow B + 1$ .  
**if**  $A \leq 40$  and  $B \leq 40$  **then**  
    **return** ACCEPT.  
**else**  
    **return** REJECT.

---

**Testing Equivalence**

The proof of Theorem 20 translates to the following explicit algorithm.

■ **Algorithm 3**  $\varepsilon$ -test for binary-alphabet EQUIVALENCE.

---

**input**  $n, \varepsilon > 0$ , two distributions  $\mu, \tau$  over  $\{0, 1\}^n$ .  
     $\mu$  is accessible through the marginal prefix oracle.  
     $\tau$  is accessible through the prefix oracle.  
**completeness** If  $\tau = \mu$ , then the output is ACCEPT with probability at least  $\frac{2}{3}$ .  
**soundness** If  $d_{\text{TV}}(\tau, \mu) > \varepsilon$ , then the output is REJECT with probability at least  $\frac{2}{3}$ .  
**let**  $\pi$  be the uniform distribution over  $[n]$ .  
**let**  $Y$  be a random variable that distributes as  $\pi \times \tau$ .  
**let**  $X$  be a random variable defined as a function of  $Y = (i, w)$ :

$$X(i, w) = \chi^2 \left( \mu_i^{x_{[i-1]=w_{[i-1]}}}(1), \tau_i^{x_{[i-1]=w_{[i-1]}}}(1) \right)$$

**let**  $\rho \leftarrow \frac{\varepsilon^2}{24 \log(\varepsilon/2n)}$ .  
**run** Levin's procedure (Algorithm 1), where its input consists of  $Y, \rho/n$ , and the black box  $((i, w), \varepsilon') \rightarrow (\text{Algorithm 2 with input } \mu_i^{x_{[i-1]=w_{[i-1]}}}, \tau_i^{x_{[i-1]=w_{[i-1]}}} \text{ and } \varepsilon')$ .  
**if** Levin's procedure accepts **then**  
    **return** ACCEPT.  
**else**  
    **return** REJECT.

---

### Testing Product

The proof of Theorem 45 translates in the binary setting to the following explicit algorithm. Note that it is almost identical to Algorithm 3, since we only substitute the marginal prefix oracle  $\tau|_i^{x^{[i-1]}=w^{[i-1]}}$  with the marginal oracle of  $\mu$ .

■ **Algorithm 4**  $\varepsilon$ -test for binary-alphabet PRODUCT.

---

**input**  $n, \varepsilon > 0$ , a distribution  $\mu$  over  $\{0, 1\}^n$ .

$\mu$  is accessible through the prefix oracle.

**completeness** If  $\mu \in \text{PRODUCT}$ , then the output is ACCEPT with probability at least  $\frac{2}{3}$ .

**soundness** If  $\min_{\tau \in \text{PRODUCT}} d_{\text{TV}}(\mu, \tau) > \varepsilon$ , then the output is REJECT with probability at least  $\frac{2}{3}$ .

**let**  $\pi$  be the uniform distribution over  $[n]$ .

**let**  $Y$  be a random variable that distributes as  $\pi \times \mu$ .

**let**  $X$  be a random variable defined as a function of  $Y = (i, w)$ :

$$X(i, w) = \chi^2(\mu|_i^{x^{[i-1]}=w^{[i-1]}}(1), \mu|_i(1))$$

**let**  $\rho \leftarrow \frac{\varepsilon^2}{24 \log(\varepsilon/2n)}$ .

**run** Levin's procedure (Algorithm 1), where its input consists of  $Y, \rho/n$ , and the black box  $((i, w), \varepsilon') \rightarrow (\text{Algorithm 2 with input } \mu|_i^{x^{[i-1]}=w^{[i-1]}}, \mu|_i \text{ and } \varepsilon')$ .

**if** Levin's procedure accepts **then**

**return** ACCEPT.

**else**

**return** REJECT.

---





# Parallelising Glauber Dynamics

Holden Lee  

Department of Applied Mathematics and Statistics, The Johns Hopkins University,  
Baltimore, MD, USA

---

## Abstract

For distributions over discrete product spaces  $\prod_{i=1}^n \Omega'_i$ , Glauber dynamics is a Markov chain that at each step, resamples a random coordinate conditioned on the other coordinates. We show that  $k$ -Glauber dynamics, which resamples a random subset of  $k$  coordinates, mixes  $k$  times faster in  $\chi^2$ -divergence, and assuming approximate tensorization of entropy, mixes  $k$  times faster in KL-divergence. We apply this to obtain parallel algorithms in two settings: (1) For the Ising model  $\mu_{J,h}(x) \propto \exp(\frac{1}{2} \langle x, Jx \rangle + \langle h, x \rangle)$  with  $\|J\| < 1 - c$  (the regime where fast mixing is known), we show that we can implement each step of  $\tilde{\Theta}(n/\|J\|_F)$ -Glauber dynamics efficiently with a parallel algorithm, resulting in a parallel algorithm with running time  $\tilde{O}(\|J\|_F) = \tilde{O}(\sqrt{n})$ . (2) For the mixed  $p$ -spin model at high enough temperature, we show that with high probability we can implement each step of  $\tilde{\Theta}(\sqrt{n})$ -Glauber dynamics efficiently and obtain running time  $\tilde{O}(\sqrt{n})$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains

**Keywords and phrases** sampling, Ising model, parallel algorithm, Markov chain, Glauber dynamics

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.49

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2307.07131>

**Acknowledgements** I want to thank Leo Du, Frederic Koehler, and Nicolas Loizou for helpful discussions.

## 1 Introduction

A key problem in computer science and statistics is to sample from a probability distribution given its probability mass function up to a constant of proportionality. The problem has been studied both over discrete spaces (such as  $\Omega^n$  for a finite set  $\Omega$ ) and continuous spaces (such as  $\mathbb{R}^n$ ); the goal is to give efficient algorithms for general classes of distributions, and in particular, to obtain optimal scaling in the dimension  $n$ . In this work we focus on minimizing the parallel running time, assuming a polynomial number of processors. In  $\mathbb{R}^n$ , it is natural to change multiple coordinates at a time using gradient-based algorithms such as Langevin dynamics and Hamiltonian Monte Carlo; many results have given algorithms that require a sublinear number of steps for log-concave distributions in various settings.

However, on discrete product spaces  $\Omega^n$ , the canonical algorithm, Glauber dynamics, involves resampling coordinates one at a time, and hence requires at least  $n$  steps in general. A natural attempt to speed up Glauber dynamics with parallel computation is to resample  $k$  coordinates at a time. We establish that under general conditions, this simple idea does indeed speed up Glauber dynamics by a factor of approximately  $k$ .

To obtain a parallel algorithm, the task remains to give a fast parallel method of resampling  $k$  coordinates. We show that this can be done in the case of the Ising model  $\mu_{J,h}$  over  $\{\pm 1\}^n$  when the interaction matrix  $J$  is bounded away from 1 in operator norm,  $\|J\| < 1 - c$ , and in the case of the mixed  $p$ -spin model at high enough temperature, both of which are known to enjoy rapid mixing of standard Glauber dynamics.



© Holden Lee;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 49; pp. 49:1–49:24



Leibniz International Proceedings in Informatics

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

The Ising model is a classical model from statistical physics which has probability mass function on  $\{\pm 1\}^n$  given by

$$\mu_{J,h}(x) = \frac{1}{Z_{J,h}} \exp\left(\frac{1}{2} \langle x, Jx \rangle + \langle h, x \rangle\right), \quad \text{where } Z_{J,h} = \sum_{x \in \{\pm 1\}^n} \exp\left(\frac{1}{2} \langle x, Jx \rangle + \langle h, x \rangle\right).$$

The regime  $\|J\| < 1$  is exactly where (based on information of the operator norm alone) Glauber dynamics is known to have fast mixing [24, 8]. To sample  $k$  coordinates, we use approximate rejection sampling with a product distribution and a further recursion for certain “bad” sets. By taking  $k = \tilde{\Theta}(n/\|J\|_F)$ , we obtain an algorithm with parallel running time  $\tilde{O}(\|J\|_F) = \tilde{O}(\sqrt{n})$ .

The **mixed  $p$ -spin model** with coefficients  $\beta_2, \beta_3, \dots$  and external field  $h \in \mathbb{R}^n$  is the random measure on  $\{\pm 1\}^n$  given by<sup>1</sup>

$$\begin{aligned} \mu_{\beta,g,h}(x) &\propto \exp(H_{\beta,g,h}(x)), \\ \text{where } H_{\beta,g,h}(x) &= \sum_{p=2}^{\infty} \frac{\beta_p \sqrt{p!}}{n^{\frac{p-1}{2}}} \sum_{1 \leq i_1 < \dots < i_p \leq n} g_{i_1, \dots, i_p} x_{i_1} \cdots x_{i_p} + \sum_{i=1}^n h_i x_i \end{aligned} \quad (1)$$

and  $g_{i_1, \dots, i_p} \sim N(0, 1)$ . By taking  $k = \Theta(\sqrt{n})$ , we obtain an algorithm with parallel running time  $\tilde{O}(\sqrt{n})$ .

## 1.1 Main results

Let  $\mu$  be a distribution on  $\prod_{i=1}^n \Omega'_i$ . We define  $k$ -Glauber dynamics as the Markov chain which given a sample  $x \in \prod_{i=1}^n \Omega'_i$ , chooses a subset  $S \subseteq [n]$  uniformly at random among subsets of size  $k$ , and resamples the coordinates in  $S$  conditional on coordinates in  $S^c$ , according to the distribution of  $\mu$ . Let  $P_{\mu,k}$  denote its Markov kernel.

We show that under general conditions,  $k$ -Glauber dynamics mixes  $k$  times faster in both  $\chi^2$  and KL-divergence. We say that a Markov kernel  $P$  with stationary distribution  $\mu$  satisfies  $\rho$ -contraction in  $\chi^2$ -divergence if  $\mathcal{D}_{\chi^2}(\nu P \|\mu) \leq \rho \mathcal{D}_{\chi^2}(\nu \|\mu)$  and similarly for  $\mathcal{D}_{\text{KL}}$ ; this can be iterated to give a mixing time bound. See Section 2.2 for background on functional inequalities (Poincaré inequality and approximate tensorization of entropy).

► **Theorem 1.1** ( *$k$ -Glauber mixes  $k$  times faster*). *Let  $\mu$  be a distribution on  $\Omega = \prod_{i=1}^n \Omega'_i$ , and let  $1 \leq k \leq n$ . Below, let  $C \geq 1$ .*

1. *If  $\mu$  satisfies a Poincaré inequality with constant  $Cn$ , then  $P_{\mu,k}$  satisfies a Poincaré inequality with constant  $O\left(\frac{Cn}{k}\right)$ , and satisfies  $(1 - \Omega\left(\frac{k}{Cn}\right))$ -contraction in  $\chi^2$ -divergence.*
2. *If  $\mu$  satisfies  $C$ -approximate tensorization of entropy (so that  $P_{\mu}$  satisfies  $(1 - \Omega\left(\frac{1}{Cn}\right))$ -contraction in KL-divergence), then  $P_{\mu,k}$  satisfies  $(1 - \Omega\left(\frac{k}{Cn}\right))$ -contraction in KL-divergence.*

Here, the  $O(\cdot)$  and  $\Omega(\cdot)$  hide only universal constants. The Poincaré inequality is equivalent to contraction in  $\chi^2$ -divergence, so part (1) gives a  $\Omega(k)$ -factor speedup to mixing in  $\chi^2$ . The analogue of the Poincaré inequality for KL is a modified log-Sobolev inequality. Although we need the slightly stronger notion of approximate tensorization of entropy to prove a speedup to mixing in KL, we note that many works that establish a modified log-Sobolev inequality do so using tensorization of entropy [12, 8]. See Section 2 for relevant background on mixing for Markov chains.

<sup>1</sup> The factor  $\sqrt{p!}$  arises as we index only over increasing sequences.

We prove Theorem 1.1 as Corollary 3.6 of the more general Theorem 3.5. We view  $k$ -Glauber dynamics as randomly erasing  $k$  coordinates one by one, and then adding them back one by one according to the right conditional distributions. This realizes  $k$ -Glauber dynamics as a composition of down and up operators  $D_{n \rightarrow n-1} \cdots D_{n-k+1 \rightarrow n-k} U_{n-k \rightarrow n-k+1} \cdots U_{n-1 \rightarrow n}$ . The assumptions give contraction of  $D_{n \rightarrow n-1}$ , and our general theorem shows that the contraction of each  $D_{j \rightarrow j-1}$  is at least as good as  $D_{n \rightarrow n-1}$  (except for an additive factor). To do this, we realize  $D_{j \rightarrow j-1}$  as  $D_{n \rightarrow n-1}$  tensorized with erasure “noise” and projected, and bound how the factor of contraction changes under these operations. We make an analogy to bounding the Poincaré and log-Sobolev constants of a distribution  $\mu$  on  $\mathbb{R}^n$  convolved with Gaussian noise, and the proximal sampler based on iteratively adding and removing Gaussian noise (more specifically, sampling from the posterior distribution given a noisy Gaussian observation of the sample from  $\mu$ ).

Algorithmically, the challenge with implementing  $k$ -Glauber dynamics is that naive enumeration for the transition kernel takes  $2^k$  time, and hence we must find a way to use the structure of the distribution to implement each step more efficiently. We show that in the case of the Ising model, we can efficiently simulate  $k$ -Glauber dynamics for  $k = \tilde{O}\left(\frac{n}{\|J^\otimes\|_F}\right)$ , to obtain a parallel algorithm running in time  $\tilde{O}\left(\|J^\otimes\|_F\right)$ , where  $J^\otimes$  denotes  $J$  with diagonal entries set to  $0^2$ . Under the assumption that  $\|J\| < 1$ , this is always at most  $\tilde{O}(\sqrt{n})$ .

► **Theorem 1.2.** *Let  $c > 0$ . With appropriate choice of constants depending only on  $c$ , if  $J$  is symmetric positive semi-definite with  $\|J\| \leq 1 - c$ , then `ParallellisingSampler` (Algorithm 1) with appropriate constants outputs a sample  $\varepsilon$ -close in TV distance from the Ising model  $\mu_{J,h}$  and, with probability at least  $1 - \varepsilon$ , runs in time  $O\left(\max\left\{\|J^\otimes\|_F, 1\right\} \text{poly log}\left(\frac{n}{\varepsilon}\right)\right)$  on a parallel machine with  $\text{poly}(n)$  processors.*

We note that our algorithm is a high-accuracy sampler: the only dependence on  $\varepsilon$  is a poly-logarithmic dependence in the running time. Notably, the number of processors does not depend on  $\varepsilon$ . We rely on the result [8] that gives optimal ( $O(n \ln n)$ ) mixing times for the Ising model for  $\|J\| < 1$  based on the theory of entropic independence.

The first attempt to implement  $k$ -Glauber dynamics is to approximate the conditional distribution of  $k$  coordinates using a carefully chosen product distribution and use rejection sampling. Using concentration results (the Hanson-Wright inequality), if  $\|J_{S \times S}^\otimes\|_F$  is small for the randomly chosen set  $S$ , then this succeeds with high probability. The complication is that  $\|J_{S \times S}^\otimes\|_F$  can sometimes be large. If this is the case, then we recurse on  $J_{S \times S}$ . By controlling the expected size of  $\|J_{S \times S}^\otimes\|_F$ , we show that the recursive calls form a subcritical branching process and with high probability, add at most a polylogarithmic overhead to the running time.

► **Theorem 1.3.** *Consider the mixed  $p$ -spin model (1). There exists an absolute constant  $\delta > 0$  such that if  $\sum_{p \geq 2} \sqrt{p^3 \ln p} \cdot \beta_p < \delta$  and  $D(\beta) = \sum_{p \geq 2} \sqrt{2^p p^3 \ln p} \cdot \beta_p < \infty$ , then with probability  $1 - \exp(-\Omega(n))$  over  $g$ , given query access to  $H_{\beta,g,h}$ , there is an algorithm which outputs a sample  $\varepsilon$ -close in TV distance from  $\mu_{\beta,g,h}$  and, with probability at least  $1 - \varepsilon$ , runs in time  $O_{D(\beta)}\left(\sqrt{n} \text{poly log}\left(\frac{n}{\varepsilon}\right)\right)$  on a parallel machine with  $\text{poly}\left(\frac{n}{\varepsilon}\right)$  processors.*

<sup>2</sup> While changing the diagonal entries of  $J$  does not change the Ising model, we need to allow  $J$  to have nonzero diagonal entries in order to be positive semi-definite.

---

**Algorithm 1** Parallel Ising Sampler (ParallellisingSampler).

---

- 1: **Input:** Interaction matrix  $J \in \mathbb{R}^{n \times n}$ , subset  $R$  of size  $m$ , external field  $h \in \mathbb{R}^R$ , error parameter  $\varepsilon \in (0, \frac{1}{2})$ .
  - 2: Let  $\varepsilon_{\text{step}} = \frac{\varepsilon}{2n^{c_4}}$ .
  - 3: **if**  $\left\| J_{R \times R}^{\otimes} \right\|_F \leq \frac{c_3}{\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right)+1}$  ( $J^{\otimes}$  denotes  $J$  with diagonal entries set to 0) **then**
  - 4:    $y \leftarrow \text{QuadraticApproxRejectionSampler} \left( H(x) = \frac{1}{2} \langle x, J_{R \times R} x \rangle + \langle h, x \rangle, \frac{c_3}{\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right)+1}, \varepsilon_{\text{step}} \right)$ .  
     (See Algorithm 2.)
  - 5: **else**
  - 6:   Let  $s = \left\lceil \frac{c_1 m}{\left(\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right)+1\right) \ln\left(\frac{n}{\varepsilon}\right) \left\| J_{R \times R}^{\otimes} \right\|_F} \right\rceil$ .
  - 7:   Let  $T = \lfloor C_2 \ln\left(\frac{n}{\varepsilon}\right) \frac{m}{s} \rfloor$ .
  - 8:   Draw  $y$  from the product distribution  $\nu_0(x) \propto e^{\langle h, x \rangle}$ .
  - 9:   **for**  $t$  from 1 to  $T$  **do**
  - 10:     Choose  $S \subseteq R$  a random subset of size  $s$ .
  - 11:      $z \leftarrow \text{ParallellisingSampler}(J, S, J_{S \times R \setminus S} y_{R \setminus S} + h_S, \varepsilon)$
  - 12:     Set  $y_S = z$ .
  - 13:   **end for**
  - 14: **end if**
  - 15: **Output:**  $y$  (Approximate sample from  $\mu_{J_{R \times R}, h}$ ).
- 

Note that a recursion is not necessary in this algorithm. Intuitively, the mean-field nature of the  $p$ -spin model ensures that with high probability all marginal distributions of  $O(\sqrt{n})$  coordinates are well-approximated by a product distribution. Though we do not investigate this further, a recursive algorithm could potentially eliminate the  $\text{poly}(1/\varepsilon)$  dependence on the number of processors as in Theorem 1.2. The proof of Theorem 1.3 is in Section 5 in the full version.

We view our result on the Ising model and the  $p$ -spin model as proofs of concept for parallelisation using  $k$ -Glauber dynamics, and hope it serves as a useful framework for constructing parallel algorithms for other families of discrete distributions. As discussed in the next section, using a different parallel algorithm, the work [35] obtains Theorem 1.2 but not Theorem 1.3.

## 1.2 Related work

We note that our Theorem 1.1 can be viewed as a complement of “local-to-global” results for mixing of the down-up walks [41, 1, 19], and is not implied by those results. Those results aim to establish mixing of Glauber dynamics (or the down-up walk) from mixing of simpler chains, while we start by assuming mixing of Glauber dynamics. In particular, [19] apply the reverse strategy: for the spin systems on graphs they consider, they show that mixing of  $\theta n$ -Glauber dynamics (for appropriate  $\theta$ ) implies mixing of Glauber dynamics.

When contraction of Glauber dynamics is derived directly from either spectral or entropic independence using local-to-global arguments, then the same arguments can be used to establish mixing of the  $k$ -Glauber (e.g., using  $k$ -uniform block factorization of entropy [19], the analogue of approximate tensorization of entropy). However, this does not apply for distributions for which mixing is established through other methods. The recent work [9]

shows that a Poincaré inequality implies spectral independence, but the bound obtained for  $k$ -Glauber through spectral independence is lossy (resulting in a power of  $n$ ). Our work can be seen as giving a general conceptual reason why mixing for Glauber must imply mixing for  $k$ -Glauber.

### 1.2.1 Continuous sampling

For log-concave distributions on  $\mathbb{R}^n$ , a long line of works on the underdamped Langevin algorithm and Metropolis-adjusted Langevin algorithm have led to high-accuracy sampling using  $\tilde{O}(n^{1/2})$  steps [3]. The randomized midpoint method for underdamped Langevin dynamics allows sampling in the weaker Wasserstein metric in  $\tilde{O}(n^{1/3})$  steps [45], and can furthermore be fully parallelised to obtain  $\varepsilon$  error with  $\text{poly}(\frac{n}{\varepsilon})$  processors. These dependencies are assuming the condition number is  $O(1)$ .

We note that the Ising model for  $\|J\| < 1$  can be decomposed as a log-concave mixture of product distributions [30, 10, 32], so these algorithms give an alternative approach to parallel algorithms for the Ising model. However, this decomposition is highly specific to the Ising model. Moreover, the Wasserstein guarantee is incompatible with a TV guarantee, and the complexity of our approach scales with  $\|J\|_F$ .

### 1.2.2 Parallel algorithms for discrete sampling

Recent work [6, 4, 7] has investigated the question of obtaining fast parallel algorithms for approximate sampling in settings where fast parallel algorithms for approximate *counting* (or computing a partition function) exist. In particular, for distributions satisfying transport stability and where the *log-Laplace* transform can be efficiently calculated (e.g., using the efficient algorithm for computing partition functions), [7] gives a  $\text{poly} \log(n/\varepsilon)$ -time algorithm with  $\text{poly}(n/\varepsilon)$  many processors (i.e., a RNC algorithm). This includes problems such as determinantal point processes and Eulerian tours. Notably they use the continuous algorithm (randomized midpoint method, discussed above) even though the problem is discrete.

In the setting of Ising models, however, we do not have a fast parallel algorithm for counting. Several works [26, 35] have studied the problem assuming the associated Dobrushin influence matrix has bounded norm. By using simultaneous updates, [35] obtains a factor- $\frac{n}{C}$  speedup for distributions whose Dobrushin influence matrix has norm bounded by  $C$ , in particular giving RNC algorithms when  $C = O(1)$  and the mixing time is  $O(n \ln n)$ . The result of [35] can also give Theorem 1.2 with a different algorithm, but cannot be used to derive Theorem 1.3. See Appendix A in the full version for details.

On the practical side, designers of Markov chain Monte Carlo algorithms in discrete spaces have taken inspiration from continuous algorithms, for example, by using gradient information to inform the proposal distribution and allow updating multiple coordinates at once [29, 50, 42]. Theoretical guarantees for these algorithms remain to be understood.

### 1.2.3 Diffusion models and the proximal sampler

Stochastic localization [23] is a measure-valued stochastic process that converges to a point mass, which is distributed according to a desired distribution  $\mu$ . As a technique, it gives a way of decomposing probability distributions that has been useful in proving functional inequalities and mixing time [17, 18], and more recently, in constructing new, *time-inhomogeneous* algorithms for sampling [22, 39].

Diffusion models [46, 47, 48] are a successful paradigm for generative modeling in machine learning, where the task is to learn and then generate samples from a distribution where only samples are given. Though the details may differ, they consist of a forward process which

adds noise to the data; reversing the process can then generate a sample from random noise. It has been observed [38] that a stochastic localization process can be viewed as the reverse process of a diffusion model.

Our analysis of  $k$ -Glauber dynamics is inspired by the analysis of the proximal sampler [34, 16, 25], which does alternating Gibbs sampling by adding Gaussian noise to the current sample, and then “de-noising” by sampling from the posterior distribution; this fits in the framework discussed above. In their analysis, [16] show that proximal sampler mixes at least as fast as Langevin in terms of  $\chi^2$  and KL-divergence. [25] show a  $\tilde{O}(n^{1/2})$  dimension dependence using a carefully chosen Gaussian proposal distribution to implement the posterior sampling step. We view the  $k$ -Glauber dynamics as a discrete analogue of the proximal Langevin algorithm, where the noise consists of erasing  $k$  coordinates, and our proof follows this analogy. In our application, we also require a careful choice of product distribution for the proposal.

## 2 Preliminaries

While many of the notions are generalizable, we will restrict ourselves to finite state spaces, and identify all measures with their probability mass functions. For more background on Markov chains, see [40].

### 2.1 Markov kernels

For finite sets  $A$  and  $B$ , a Markov kernel  $K$  from  $A$  to  $B$  is a function  $A \times B \rightarrow \mathbb{R}_{\geq 0}$  or equivalently, a matrix  $\mathbb{R}_{\geq 0}^{A \times B}$ , where the rows sum to 1. If  $\mu$  is a measure on  $A$ , then  $\mu K$  is a measure on  $B$ ; if  $f$  is a function  $B \rightarrow \mathbb{R}$ , then  $Kf$  is a function  $A \rightarrow \mathbb{R}$ ; these correspond to matrix-vector multiplication. Composition of kernels  $K_1$  from  $A$  to  $B$  and  $K_2$  from  $B$  to  $C$  gives a kernel  $K_1 K_2$  from  $A$  to  $C$ , which corresponds to matrix multiplication. For  $f, g$  functions on  $A$  and  $\mu$  a measure on  $A$ , let  $\langle f, g \rangle_\mu = \sum_{x \in A} \mu(x) f(x) g(x)$ . For a kernel  $K : A \times B \rightarrow \mathbb{R}_{\geq 0}$ , given measures  $\mu_1, \mu_2$  on  $A$  and  $B$  respectively, we think of  $K$  as a linear map  $L^2(\mu_1) \rightarrow L^2(\mu_2)$ ; then its adjoint  $K^* : B \times A \rightarrow \mathbb{R}_{\geq 0}$  is a linear map  $L^2(\mu_2) \rightarrow L^2(\mu_1)$  satisfying  $\langle f, Kg \rangle_{\mu_1} = \langle K^* f, g \rangle_{\mu_2}$  for any  $f \in L^2(\mu_1), g \in L^2(\mu_2)$ .

► **Definition 2.1.**  *$k$ -Glauber dynamics with stationary distribution  $\mu$  on  $\Omega$  is the Markov chain where at each step, if the current sample is  $x$ , we choose a subset  $S$  uniformly at random in  $\binom{\Omega}{k}$  (subsets of size  $k$ ), and resample the coordinates in  $S$  according to  $\mu(X_S | X_{S^c} = x_{S^c})$ . Let  $P_{\mu, k}$  denote the transition operator. For  $k = 1$ , we simply call it Glauber dynamics, and let  $P_\mu$  denote the Markov kernel.*

► **Definition 2.2.** *Let  $0 \leq \ell \leq k \leq n$ . Let  $\mu$  be a distribution on  $\binom{[n]}{k}$ . Define the **down operator**  $D_{k \rightarrow \ell}$  and **up operator**  $U_{\ell \rightarrow k}$  as Markov kernels  $\binom{[n]}{k} \times \binom{[n]}{\ell} \rightarrow \mathbb{R}_{\geq 0}$  and  $\binom{[n]}{\ell} \times \binom{[n]}{k} \rightarrow \mathbb{R}_{\geq 0}$ , respectively, with*

$$D_{k \rightarrow \ell}(A, B) = \mathbb{1}_{B \subseteq A} \frac{1}{\binom{k}{\ell}} \quad U_{\ell \rightarrow k}(B, A) = \mathbb{1}_{B \subseteq A} \frac{\mu(A)}{\sum_{A' \supseteq B} \mu(A')}.$$

Let  $\mu_\ell = \mu D_{k \rightarrow \ell}$  for  $0 \leq \ell \leq k$ , and define the  $k \leftrightarrow \ell$  **down-up walk** and  $\ell \leftrightarrow k$  **up-down walk** by

$$P_{k \leftrightarrow \ell}^\nabla = D_{k \rightarrow \ell} U_{\ell \rightarrow k} \quad P_{\ell \leftrightarrow k}^\Delta = U_{\ell \rightarrow k} D_{k \rightarrow \ell}.$$

Note that  $D_{k \rightarrow \ell}$  does not depend on  $\mu$  while  $U_{\ell \rightarrow k}$  does; we suppress the dependency in the notation. Note that  $D_{k \rightarrow \ell} D_{\ell \rightarrow m} = D_{k \rightarrow m}$  and  $U_{m \rightarrow \ell} U_{\ell \rightarrow k} = U_{m \rightarrow k}$ . As operators,  $D_{k \rightarrow \ell} : L^2(\mu_\ell) \rightarrow L^2(\mu_k)$  and  $U_{\ell \rightarrow k} : L^2(\mu_k) \rightarrow L^2(\mu_\ell)$  are adjoint.

► **Definition 2.3.** Let  $\mu$  be a measure on  $\Omega' = \Omega'_1 \times \cdots \times \Omega'_n$ . Define the **homogenization** of  $\mu$  to be the measure  $\mu^{\text{hom}}$  over  $\binom{\Omega}{n}$ , where  $\Omega = \bigcup_{i=1}^n \Omega'_i \times \{i\}$  and  $\sigma \in \Omega'$  is identified with  $\{(\sigma_1, 1), \dots, (\sigma_n, n)\}$ . (For short, we will write  $\Omega = \bigsqcup_{i=1}^n \Omega'_i$  in the following.) For any property  $\mathcal{P}$  defined for measures  $\binom{\Omega}{n}$ , we say that  $\mu$  satisfies  $\mathcal{P}$  if  $\mu^{\text{hom}}$  satisfies  $\mathcal{P}$ .

Under this identification,  $k$ -Glauber dynamics corresponds to the  $n \leftrightarrow n - k$  down-up walk, as the down step corresponds to erasing  $k$  coordinates and the up step corresponds to restoring them with the correct conditional probabilities.

## 2.2 Functional inequalities

► **Definition 2.4.** Let  $M = (\Omega, P)$  be an ergodic, reversible Markov chain with stationary distribution  $\mu$ . Define the associated Dirichlet form as the inner product

$$\mathcal{E}_P(f, g) = \langle f, (I - P)g \rangle_\mu = \frac{1}{2} \sum_{x, y \in \Omega} \mu(x) P(x, y) (f(x) - f(y))(g(x) - g(y))$$

When  $\mu$  is a distribution on  $\Omega = \prod_{i=1}^n \Omega'_i$ , we write  $\mathcal{E}_\mu = \mathcal{E}_{P_\mu}$ ; we will similarly make other such replacements without comment.

► **Definition 2.5.** Keeping the assumptions above, we say that  $P$  satisfies a **Poincaré inequality** with constant  $C$  if for all  $f : \Omega \rightarrow \mathbb{R}$ ,

$$\text{Var}_\mu(f) \leq C \mathcal{E}_P(f, f).$$

We say  $\mu$  satisfies a Poincaré inequality with constant  $C$  if the Glauber dynamics with stationary distribution  $\mu$ ,  $P_\mu$ , satisfies a Poincaré inequality with constant  $C$ .

When  $P$  is self-adjoint ( $M$  is reversible), this is the same as saying that  $\lambda_2(P) \leq 1 - \frac{1}{C}$ , where  $\lambda_k(\cdot)$  denotes the  $k$ th largest eigenvalue.

► **Definition 2.6.** Let  $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$  be a strictly convex function with  $f(1) = 0$ . For measures  $\nu \ll \mu$  on  $\Omega$ , define the  **$f$ -divergence** by

$$\mathcal{D}_f(\nu \parallel \mu) = \mathbb{E}_{x \sim \mu} f\left(\frac{\nu(x)}{\mu(x)}\right).$$

In particular, define the  $\chi^2$  and KL-divergences by  $\mathcal{D}_{\chi^2} = D_{(x-1)^2}$  and  $\mathcal{D}_{\text{KL}} = D_{x \ln x}$ .

► **Definition 2.7.** We say that Markov kernel  $P : \Omega_1 \times \Omega_2 \rightarrow \mathbb{R}$  satisfies  **$\rho$ -contraction in  $f$ -divergence** with respect to  $\mu_1$  if for all  $\nu_1 \ll \mu_1$ ,

$$\mathcal{D}_f(\nu_1 P \parallel \mu_1 P) \leq \rho \mathcal{D}_f(\nu_1 \parallel \mu_1).$$

Contraction in  $\chi^2$  and KL-divergence is also referred to as variance or entropy contraction, respectively.

► **Proposition 2.8.** Let  $P : \Omega_1 \times \Omega_2 \rightarrow \mathbb{R}_{\geq 0}$  be a Markov kernel. The following are equivalent, for  $C \leq 1$ :

1.  $P$  satisfies  $(1 - C)^2$ -contraction in  $\chi^2$ -divergence with respect to  $\mu$ .
2. For all  $f : \Omega_1 \rightarrow \mathbb{R}$ ,

$$\text{Var}_{\mu P}(Pf) \leq (1 - C)^2 \text{Var}_\mu(f).$$

3. (For  $\Omega_1 = \Omega_2$ ,  $P$  reversible)  $P$  satisfies a Poincaré inequality with constant  $\frac{1}{C}$ .
  4. (For  $P$  of the form  $P = DD^*$ , e.g.,  $P_{k \leftrightarrow k-1}^\nabla = D_{k \rightarrow k-1} U_{k-1 \rightarrow k}$ )  $D$  satisfies  $(1 - C)$ -contraction in  $\chi^2$ -divergence.
  5. (For  $P = DD^*$ )  $D^*$  satisfies  $(1 - C)$ -contraction in  $\chi^2$ -divergence.
- Here, the adjoint is with respect to the measures  $\mu$  and  $\mu D$ .

**Proof sketch.** See full version. ◀

► **Definition 2.9.** A measure  $\mu$  on  $\binom{[n]}{k}$  satisfies *C-approximate tensorization of entropy* if  $D_{k \rightarrow k-1}$  satisfies  $(1 - \frac{1}{Ck})$ -contraction in KL-divergence, i.e., for any  $\nu \ll \mu$ ,

$$\mathcal{D}_{\text{KL}}(\nu D_{k \rightarrow k-1} \| \mu D_{k \rightarrow k-1}) \leq \left(1 - \frac{1}{Ck}\right) \mathcal{D}_{\text{KL}}(\nu \| \mu).$$

We have the following alternate characterization for a measure defined on a product space. Define the entropy of a function  $f$  on a probability space by  $\text{Ent}_\mu[f] = \mathbb{E}_\mu[f \ln f] - \mathbb{E}_\mu[f] \ln \mathbb{E}_\mu[f]$ .

► **Proposition 2.10** ([19, Lemma 2.7]). Let  $\mu$  be a measure on  $\Omega = \Omega'_1 \times \cdots \times \Omega'_n$ . Then  $\mu$  satisfies *C-approximate tensorization of entropy* iff for all  $f : \Omega \rightarrow \mathbb{R}_{\geq 0}$ ,

$$\text{Ent}_\mu[f] \leq C \sum_{k=1}^n \mathbb{E}_\mu \left[ \text{Ent}_{\mu(X_k = \cdot | X_{\sim k} = x_{\sim k})}[f] \right],$$

where  $\sim k$  denotes the coordinates besides  $k$ .

► **Remark 2.11.** Proposition 2.8 shows that for contraction in  $\chi^2$ -divergence, nothing is lost if we consider  $P_{k \leftrightarrow k-1}^\nabla$  or  $D_{k \rightarrow k-1}$ ,  $U_{k-1 \rightarrow k}$  separately. However, the distinction is important for KL, as contraction of  $P_{k \leftrightarrow k-1}^\nabla$  may not imply contraction of  $D_{k \rightarrow k-1}$  or  $U_{k-1 \rightarrow k}$  separately; hence the definition of approximate tensorization of entropy. Approximate tensorization of entropy is stronger than the modified log-Sobolev inequality (which implies mixing for  $P_{k \leftrightarrow k-1}^\nabla$ ), but weaker than the log-Sobolev inequality.

### 2.3 Additional notation

For  $f : \prod_{i=1}^n \Omega'_i \rightarrow \mathbb{R}$ , and  $x \in \prod_{i \in S^c} \Omega'_i$ , define the restriction  $f_x : \prod_{i \in S} \Omega'_i$  by  $f_x(y) = f(x, y)$  with  $(x, y)$  treated as an element of  $\prod_{i=1}^n \Omega'_i$ .

Let  $x_{i \leftarrow b}$  denote  $x$  with  $x_i$  set to  $b$ . For  $f : \{\pm 1\}^n$ , let  $D_i f(x) := \frac{1}{2}[f(x_{i \leftarrow 1}) - f(x_{i \leftarrow -1})]$  and define  $\nabla f : \{\pm 1\}^n \rightarrow \mathbb{R}^n$  by

$$\nabla f(x) = (D_1 f(x), \dots, D_n f(x))$$

and  $\nabla^2 f : \{\pm 1\}^n \rightarrow \mathbb{R}^{n \times n}$  by  $(\nabla^2 f(x))_{i,j} = D_i D_j f(x)$  (note  $(\nabla^2 f(x))_{i,i} = 0$ ).

For  $x \in \{\pm 1\}^n$  and  $S \subseteq [n]$ , let  $x^S$  denote  $\prod_{i \in S} x_i$ . For a function  $f : \{\pm 1\}^n \rightarrow \mathbb{R}$ , we denote the degree  $d$  part of  $f$  by  $f^{(d)}$ , and define  $f^{\geq d} = \sum_{p \geq d} f^{(p)}$ , etc., so that we have the decomposition

$$f(x) = \sum_{p=0}^n f^{(p)}(x) = \sum_{p=0}^n \sum_{|I|=p} a_I x^I$$

for some coefficients  $a_I$ . We take  $f_x^{(d)}$  to mean that we take the restriction first and then the degree- $d$  part.

For a scalar-valued function  $f$  and  $x \in \mathbb{R}^n$ , we let  $f(x)$  denote coordinate-wise evaluation.



### 3 $k$ -Glauber mixes $k$ times as fast

To show that  $k$ -Glauber mixes  $k$  times more quickly than Glauber dynamics, write  $P_\mu = D_{n \rightarrow n-1} U_{n-1 \rightarrow n}$  and  $P_{\mu,k} = D_{n \rightarrow n-1} \cdots D_{n-k+1 \rightarrow n-k} U_{n-k \rightarrow n-k+1} \cdots U_{n-1 \rightarrow n}$ ; the task is then to show that  $D_{j \rightarrow j-1}$ ,  $j \leq n$  are roughly at least as contractive as  $D_{n \rightarrow n-1}$ . We note that this is like the reverse of the usual “local-to-global” argument for high-dimensional expanders [1] (and will be easier!). We also note this approach relates to inductive arguments in prior work (e.g., [20, Lemma 11]).

Viewing the problem in this way, we note the similarity to the proximal sampler [16], each step of which involves adding and removing Gaussian noise; they bound the contraction in  $\chi^2$  and KL-divergence of this process based on the Poincaré and log-Sobolev constants of the original distribution.

As inspiration, we first recall the following fact, which bounds the Poincaré or log-Sobolev constant of a convolution of two measures on  $\mathbb{R}^n$ . (The convolution  $\mu_1 * \mu_2$  is defined as the distribution of  $X + Y$  where  $X \sim \mu_1$  and  $Y \sim \mu_2$  are independent.) As we will not cover the theory of functional inequalities over  $\mathbb{R}^n$ , this is meant only as a suggestion of how we might proceed. For details and generalizations, see [14].

► **Lemma.** *Suppose that  $\mu_1, \mu_2$  are distributions on  $\mathbb{R}^n$  with Poincaré constants  $C_1, C_2$ , respectively. Then  $\mu_1 * \mu_2$  has Poincaré constant bounded by  $C_1 + C_2$ . The same holds true for the log-Sobolev constant. In particular, this holds true for  $\mu_2$  being a Gaussian of variance  $C_1$ .*

**Proof sketch.** Let  $M_m$  denote multiplication by  $m$ . Then  $\mu_i M_{m_i}^{-1}$  has Poincaré constant  $m_i^2 C_i$ . By tensorization,  $\mu = \mu_1 M_{m_1}^{-1} \otimes \mu_2 M_{m_2}^{-1}$  has Poincaré constant  $\max\{C_1 m_1^2, C_2 m_2^2\}$ . Consider the projection  $\pi(x_1, x_2) = \frac{x_1}{m_1} + \frac{x_2}{m_2}$ . When  $\frac{1}{m_1} + \frac{1}{m_2} = 1$ , this can be realized as projection onto the vector  $(\frac{1}{m_1}, \frac{1}{m_2})$ , so the Poincaré constant does not increase:  $C_P(\mu \pi^{-1}) \leq C_P(\mu)$ . Note that we exactly have  $\mu \pi^{-1} = \mu_1 * \mu_2$ . Choosing  $\frac{1}{m_1} = \frac{C_1}{C_1 + C_2}$  and  $\frac{1}{m_2} = \frac{C_2}{C_1 + C_2}$  gives the bound. The same argument works for the log-Sobolev constant. ◀

We will carry out the same steps as in this proof: define a tensorization operation which preserves the Poincaré or approximate tensorization of entropy constant, and a projection which can only improve it. As in the sketch above, it will help to weight the components appropriately in the tensorization step.

To obtain the  $D_{k \rightarrow k-1}$  operator from the  $D_{n \rightarrow n-1}$  operator, we will tensorize with the appropriate “noise” distribution, which in this case is that of erasing  $n - k$  coordinates. We note that while bit-flip noise is the more natural analogue of gaussian noise, the “denoising” step is more difficult to implement, while erasure noise connects more nicely with existing notions.

### 3.1 Tensorization and projection

The following proposition is similar to classical results on preservation of functional inequalities under tensorization, which corresponds to taking a product of Markov chains. However, we need to work with operators between different spaces to obtain contraction for the down operator, so we need the result given in Proposition 3.1. As in Remark 2.11, nothing would be lost for  $\chi^2$ -divergence if we considered the down-up walk – so we could use existing results – but for KL we need to bound contraction for just the down operator. This is important because we aim to bound contraction for  $P_{n \leftrightarrow n-k}^\nabla = D_{n \rightarrow n-1} \cdots D_{n-k+1 \rightarrow n-k} U_{n-k \rightarrow n-k+1} \cdots U_{n-1 \rightarrow n}$ , and in the case of KL, we cannot obtain this by bounding contraction for just operators of the form  $P_{k \leftrightarrow k-1}^\nabla = D_{k \rightarrow k-1} U_{k-1 \rightarrow k}$ .

► **Proposition 3.1** (Contraction under tensorization). *Suppose that  $P_i$  is a Markov kernel from  $\Omega_i$  to  $\Omega'_i$ , for  $i = 1, 2$ . Let  $I_i$  denote the identity kernel on  $\Omega_i$  and define  $P = p(P_1 \otimes I_2) + (1-p)(I_1 \otimes P_2)$  as a Markov kernel from  $\Omega_1 \times \Omega_2$  to  $(\Omega'_1 \times \Omega_2) \sqcup (\Omega_1 \times \Omega'_2)$ .*

*Let  $\mathcal{D}_f = \mathcal{D}_{\text{KL}}$  or  $\mathcal{D}_{\chi^2}$ . If  $P_i$  satisfies  $(1 - \kappa_i)$ -contraction in  $f$ -divergence with respect to  $\mu_i$ , then  $P$  satisfies  $(1 - \kappa)$ -contraction in  $f$ -divergence with respect to  $\mu_1 \otimes \mu_2$ , where  $\kappa = \min\{p\kappa_1, (1-p)\kappa_2\}$ . In particular, if  $p = \frac{\kappa_2}{\kappa_1 + \kappa_2}$ , then  $\kappa = \frac{\kappa_1 \kappa_2}{\kappa_1 + \kappa_2} = \frac{1}{\kappa_1^{-1} + \kappa_2^{-1}}$ .*

**Proof.** First consider KL-divergence. Let  $\nu$  be a measure on  $\Omega_1 \times \Omega_2$ . Let  $\Pi_i$  and  $\Pi'_i$  denote the projection kernels to  $\Omega_i$  and  $\Omega'_i$  respectively, for  $i = 1, 2$  (i.e., taking marginals). We calculate

$$\begin{aligned} B &:= \mathcal{D}_{\text{KL}}(p\nu(P_1 \otimes I_2) + (1-p)\nu(I_1 \otimes P_2) \| p(\mu'_1 \times \mu_2) + (1-p)(\mu_1 \times \mu'_2)) \\ &= p\mathcal{D}_{\text{KL}}(\nu(P_1 \otimes I_2) \| \mu'_1 \times \mu_2) + (1-p)\mathcal{D}_{\text{KL}}(\nu(I_1 \otimes P_2) \| \mu_1 \times \mu'_2), \end{aligned} \quad (2)$$

since the spaces  $\Omega'_1 \times \Omega_2$  and  $\Omega_1 \times \Omega'_2$  are disjoint. Now, by the chain rule of KL-divergence and entropy contraction,

$$\begin{aligned} \mathcal{D}_{\text{KL}}(\nu(P_1 \otimes I_2) \| \mu'_1 \times \mu_2) &= \mathbb{E}_{x_2 \sim \mu_2} \mathcal{D}_{\text{KL}}((\nu(P_1 \otimes I_2))(\cdot | x_2) \| \mu'_1) + \mathcal{D}_{\text{KL}}(\nu(P_1 \otimes I_2) \Pi_2 \| \mu_2) \\ &= \mathbb{E}_{x_2 \sim \mu_2} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_2) P_1 \| \mu'_1) + \mathcal{D}_{\text{KL}}(\nu \Pi_2 \| \mu_2) \\ &\leq (1 - \kappa_1) \mathbb{E}_{x_2 \sim \mu_2} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_2) \| \mu_1) + \mathcal{D}_{\text{KL}}(\nu \Pi_2 \| \mu_2). \end{aligned}$$

By symmetry, the same calculation holds for the second term in (2), giving us

$$\begin{aligned} B &\leq p[(1 - \kappa_1) \mathbb{E}_{x_2 \sim \mu_2} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_2) \| \mu_1) + \mathcal{D}_{\text{KL}}(\nu \Pi_2 \| \mu_2)] \\ &\quad + (1-p)[(1 - \kappa_2) \mathbb{E}_{x_1 \sim \mu_1} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_1) \| \mu_2) + \mathcal{D}_{\text{KL}}(\nu \Pi_1 \| \mu_1)]. \end{aligned}$$

We wish to compare this with

$$\begin{aligned} A &:= \mathcal{D}_{\text{KL}}(\mu \| \mu_1 \times \mu_2) \\ &= \underbrace{\mathbb{E}_{x_2 \sim \mu_2} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_2) \| \mu_1)}_{C_{2,1}} + \underbrace{\mathcal{D}_{\text{KL}}(\nu \Pi_2 \| \mu_2)}_{C_2} \\ &= \underbrace{\mathbb{E}_{x_1 \sim \mu_1} \mathcal{D}_{\text{KL}}(\nu(\cdot | x_1) \| \mu_2)}_{C_{1,2}} + \underbrace{\mathcal{D}_{\text{KL}}(\nu \Pi_1 \| \mu_1)}_{C_1}. \end{aligned}$$

By convexity of KL-divergence,  $C_{1,2} \geq C_2$  and  $C_{2,1} \geq C_1$ . Hence

$$\begin{aligned} B &\leq p((1 - \kappa_1)C_{2,1} + C_2) + (1-p)((1 - \kappa_2)C_{1,2} + C_1) \\ &\leq \max\{(1-p)(1 - \kappa_2) + p, p(1 - \kappa_1) + (1-p)\} [p(C_{2,1} + C_2) + (1-p)(C_{1,2} + C_1)] \\ &= (1 - \min\{(1-p)\kappa_2, p\kappa_1\}) A. \end{aligned}$$

Next consider  $\chi^2$ -divergence. It suffices to prove the following equivalent statement on contraction of variance (Proposition 2.8): for any  $f : \Omega_1 \times \Omega_2 \rightarrow \mathbb{R}$ , considering  $P^* = p(P_1^* \otimes I_2) \oplus (1-p)(I_1 \otimes P_2^*)$ , we have

$$\text{Var}_{\mu P}(P^* f) \leq (1 - \kappa) \text{Var}_{\mu}(f),$$

which is equivalent to  $\sigma_2(P^*) \leq 1 - \kappa$  and hence to  $\lambda_2(PP^*)^2 = \sigma_2(PP^*)^2 \leq (1 - \kappa)^2$ . Now  $PP^* = p(P_1 P_1^* \otimes I_2) + (1-p)(I_1 \otimes P_2 P_2^*)$  is exactly the transition matrix of the weighted product of two Markov chains with  $\lambda_2(P_i P_i^*) \leq 1 - \kappa_i$ , so it is well-known that

$$\lambda_2(PP^*) \leq \max\{p + (1-p)(1 - \kappa_2), (1-p) + p(1 - \kappa_1)\} = 1 - \max\{(1-p)\kappa_2, p\kappa_1\}.$$

(A quick way to see this is as follows: if  $\{f_i\}$  are the eigenvectors of  $P_1 P_1^*$  and  $\{g_j\}$  are the eigenvectors of  $P_2 P_2^*$ , then  $\{f_i g_j\}$  are the eigenvectors of  $P^* P$ , and the second largest eigenvalue is when  $f_i = 1$  or  $g_j = 1$ .) ◀

Though we do not do it here, the proofs can be put on the same footing and generalized using the notion of  $f$ -entropy [14].

► **Proposition 3.2** (Contraction under projection). *Suppose that  $P$  is a Markov kernel from  $\Omega_1$  to  $\Omega_2$ , and  $\mu_1, \mu_2$  are measures on  $\Omega_1, \Omega_2$  such that  $\mu_1 P = \mu_2$ . Let  $\pi_i : \Omega_i \rightarrow \Omega'_i$  be maps and  $\mu'_i = \mu_i \pi_i^{-1}$ . Define the projected Markov kernel  $P' : \Omega'_1 \times \Omega'_2 \rightarrow \mathbb{R}_{\geq 0}$  by*

$$P'(x'_1, x'_2) = \sum_{\substack{x_1 \in \Omega_1 \\ \pi_1(x_1) = x'_1}} \sum_{\substack{x_2 \in \Omega_2 \\ \pi_2(x_2) = x'_2}} \mu_1(x_1 | \pi(x_1) = x'_1) P(x_1, x_2).$$

If  $P$  satisfies  $\rho$ -contraction in  $f$ -divergence with respect to  $\mu_1$ , then  $P'$  also satisfies  $\rho$ -contraction in  $f$ -divergence with respect to  $\mu'_1$ .

In words, in the projected Markov chain, given  $x'_1$ , we draw  $x_1$  projecting to  $x'_1$  from the “prior”, move according to  $P$ , and then project back down; then the projected kernel always has at least as much contraction as the original one. See e.g., [37] for the statement for the Poincaré constant.

**Proof.** Let  $\nu'_1 \ll \mu'_1$  be a measure on  $\Omega'_1$ . Define  $\nu_1(x) = \nu'_1(\pi_1(x)) \mu_1(X = x | \pi_1(X) = \pi_1(x))$ . Then  $\mathbb{E}_{\mu_1(\cdot | \pi_1(X) = x'_1)} f\left(\frac{\nu_1(x)}{\mu_1(x)}\right) = \frac{\nu'_1(x'_1)}{\mu'_1(x'_1)}$ , so

$$\mathcal{D}_f(\nu'_1 \| \mu'_1) = \mathbb{E}_{x'_1 \sim \mu'_1} f\left(\frac{\nu'_1(x'_1)}{\mu'_1(x'_1)}\right) = \mathcal{D}_f(\nu_1 \| \mu_1).$$

By contraction of  $P$  and the data processing inequality,

$$\mathcal{D}_f((\nu_1 P) \pi_2^{-1} \| \mu'_2) \leq \mathcal{D}_f(\nu_1 P \| \mu_2) \leq \rho \mathcal{D}_f(\nu_1 \| \mu_1) = \rho \mathcal{D}_f(\nu'_1 \| \mu'_1).$$

Finally, note that  $(\nu_1 P) \pi_2^{-1} = \nu'_1 P'$  by definition of  $P'$ . ◀

### 3.2 Contraction improves going down

We now show that for  $k < n$ , contraction in KL and  $\chi^2$  for  $D_{k \rightarrow k-1}$  will only be better than contraction of  $D_{n \rightarrow n-1}$ , except up to an additive constant.

► **Lemma 3.3.** *Let  $\mu$  be the uniform distribution on  $\binom{[n]}{k}$ . Then  $D_{k \rightarrow k-1}$  has  $(1 - \frac{1}{k})$ -contraction in KL, and  $\mu$  satisfies 1-approximate tensorization of entropy. Moreover,  $D_{k \rightarrow k-1}$  and  $U_{k-1 \rightarrow k}$  satisfy  $(1 - \frac{n}{k(n-k+1)})$ -contraction in  $\chi^2$ .*

We note that the down-up walk on the uniform distribution on  $\binom{[n]}{k}$  is a rescaling of the Bernoulli-Laplace diffusion model, for which mixing and functional inequalities have been extensively studied [21, 33, 27, 13, 43], and we have not attempted to find the best result for KL.

**Proof.** Note that  $\mu$  is log-concave, by the results of [5] and the fact that  $\binom{[n]}{k}$  is a matroid. Then 1-approximate tensorization of entropy follows from [8, Theorem 5].

To note the improved bound for  $\chi^2$ , note first that the probability of staying at the same set is  $\frac{1}{n-k+1}$ , so  $P_{k \leftrightarrow k-1}^\nabla = \frac{1}{n-k+1} I + \frac{n-k}{n-k+1} P_{k \leftrightarrow k-1}^\vee$  where  $P_{k \leftrightarrow k-1}^\vee$  is the “non-lazy” walk which swaps an occupied and non-occupied space at random. Hence  $I - P_{k \leftrightarrow k-1}^\nabla = \frac{n-k}{n-k+1} (I - P_{k \leftrightarrow k-1}^\vee)$ . This in turn satisfies

$$I - P_{k \leftrightarrow k-1}^\nabla = \frac{n(n-1)/2}{k(n-k)} \cdot \frac{2}{n-1} \cdot L$$

## 49:12 Parallelising Glauber Dynamics

where  $L$  is the generator of the Bernoulli-Laplace diffusion model with parameters  $(n, k)$  (a random transposition occurs with rate 1). Here, the  $\frac{2}{n-1}$  is the scaling factor to convert to a discrete-time walk and  $\frac{n(n-1)/2}{k(n-k)}$  takes into account that we are randomly choosing from  $k(n-k)$  transpositions that move a particle to an unoccupied space, rather than an arbitrary transposition. By [43],  $\lambda_{\min}(L) = 1$ . Hence the spectral gap for  $P_{k \leftrightarrow k-1}^{\nabla}$  is

$$\frac{n-k}{n-k+1} \cdot \frac{n(n-1)/2}{k(n-k)} \cdot \frac{2}{n-1} = \frac{n}{k(n-k+1)}. \quad \blacktriangleleft$$

► **Lemma 3.4.** *Let  $\mu$  be a distribution on  $\prod_{i=1}^n \Omega'_i$  and  $\mu^{\text{hom}}$  its homogenization on  $\binom{\Omega}{n}$ , where  $\Omega = \bigsqcup_{i=1}^n \Omega'_i$ , and  $\mu_m = \mu^{\text{hom}} D_{n \rightarrow m}$ . Let  $\mathcal{D}_f = \mathcal{D}_{\text{KL}}$  or  $\mathcal{D}_{\chi^2}$ . For each  $k$ , let  $\kappa_k$  be the largest number such that  $D_k := D_{k \rightarrow k-1}$  satisfies  $(1 - \kappa_k)$ -contraction in  $f$ -divergence with respect to  $\mu_k$ . Suppose that for the uniform distribution  $\binom{[n]}{k}$  that  $D_{k \rightarrow k-1}$  satisfies  $(1 - \kappa_{\text{BL},k})$ -contraction in  $f$ -divergence. Then*

$$\kappa_k \geq \frac{n\kappa_n \kappa_{\text{BL},k}}{n\kappa_n + k\kappa_{\text{BL},k}} \geq \begin{cases} \frac{n\kappa_n}{k((n-k+1)\kappa_n+1)}, & \mathcal{D}_f = \mathcal{D}_{\chi^2} \\ \frac{n\kappa_n}{k(n\kappa_n+1)}, & \mathcal{D}_f = \mathcal{D}_{\text{KL}}. \end{cases}$$

**Proof.** Consider the kernel  $P = p(D_n \otimes I_{\binom{[n]}{k}}) + (1-p)(I_{\binom{\Omega}{n}} \otimes D_k)$  from  $\binom{\Omega}{n} \times \binom{[n]}{k}$  to  $\binom{\Omega}{n-1} \times \binom{[n]}{k} \cup \binom{\Omega}{n} \times \binom{[n]}{k-1}$ , where  $D_k$  denotes the down operator  $\binom{[n]}{k} \times \binom{[n]}{k-1} \rightarrow \mathbb{R}_{\geq 0}$ . By tensorization (Proposition 3.1), for  $p = \frac{\kappa_{\text{BL},k}}{\kappa_n + \kappa_{\text{BL},k}}$  and  $\kappa = \frac{1}{\kappa_n^{-1} + \kappa_{\text{BL},k}^{-1}}$ ,  $P$  satisfies  $(1 - \kappa)$ -contraction in  $f$ -divergence with respect to  $\mu_n \otimes \text{Uniform}\left(\binom{[n]}{k}\right)$ . Define the projections  $\pi_1 : \binom{\Omega}{n} \times \binom{[n]}{k} \rightarrow \binom{\Omega}{k}$  and  $\pi_2 : \binom{\Omega}{n-1} \times \binom{[n]}{k} \cup \binom{\Omega}{n} \times \binom{[n]}{k-1} \rightarrow \binom{\Omega}{k-1} \cup \binom{\Omega}{k}$  both as

$$\pi(S, A) = S \cap \bigcup_{i \in A} \Omega_i,$$

i.e., if  $S$  corresponds to  $x \in \prod_{i=1}^n \Omega'_i$ , we keep only the coordinates in the set  $A \in \binom{[n]}{k}$ .

We claim the projected kernel as defined in Proposition 3.2 is

$$P' = \underbrace{\left(1 - \frac{p(n-k)}{n}\right)}_{\text{(I)}} D_k + \underbrace{\frac{p(n-k)}{n} I}_{\text{(II)}}$$

from  $\binom{\Omega}{k}$  to  $\binom{\Omega}{k-1} \cup \binom{\Omega}{k}$ . To see this, we first identify  $x_A \in \prod_{i=1}^{i \in A} \Omega'_i$  with its homogenization in  $\binom{\Omega}{|A|}$  (Definition 2.3). Then  $\mu_1(\cdot | \pi_1(x_1) = x_A)$  is the distribution of  $(x, A)$  where  $x_{A^c}$  is distributed as  $\mu(X_{A^c} = x_{A^c} | X_A = x_A)$ . Under the transition  $P$ :

1. With probability  $p$ , we remove a coordinate  $i$  of  $x$ . In this case,
  - a. with probability  $\frac{k}{n}$ ,  $i \in A$ , and projection by  $\pi_2$  then gives  $x_{A \setminus \{i\}}$  for a random index  $i$ .
  - b. with probability  $\frac{n-k}{n}$ ,  $i \notin A$ , and projection by  $\pi_2$ .
2. With probability  $1-p$ , we remove an element  $i$  of  $A$ , and projection gives  $x_{A \setminus \{i\}}$  for a random  $i \in A$ .

Then (1a) and (2) give the term (I) and (1b) gives the term (II).

Because

$$\begin{aligned} \mathcal{D}_f \left( \nu P' \parallel \left(1 - \frac{p(n-k)}{n}\right) \mu_{k-1} + \frac{p(n-k)}{n} \mu_k \right) \\ = \left(1 - \frac{p(n-k)}{n}\right) \mathcal{D}_{\text{KL}}(\nu D_k \parallel \mu_{k-1}) + \frac{p(n-k)}{n} \mathcal{D}_{\text{KL}}(\nu \parallel \mu_k) \end{aligned}$$

(the component measures have disjoint support), we have that  $D_k$  satisfies  $(1 - \kappa_k)$ -contraction in  $f$ -divergence iff  $P'$  satisfies  $(1 - \kappa_k (1 - \frac{p(n-k)}{n}))$ -contraction in  $f$ -divergence. Proposition 3.2 then gives us

$$\begin{aligned} \kappa_k \left( 1 - \frac{\kappa_{\text{BL},k}}{\kappa_n + \kappa_{\text{BL},k}} \cdot \frac{n-k}{n} \right) &\geq \frac{1}{\kappa_n^{-1} + \kappa_{\text{BL},k}^{-1}} \\ \implies \kappa_k &\geq \frac{n\kappa_n\kappa_{\text{BL},k}}{n\kappa_n + k\kappa_{\text{BL},k}}. \end{aligned}$$

Plugging in the result of Lemma 3.3 then gives the result.  $\blacktriangleleft$

**► Theorem 3.5.** *Let  $\mu$  be a distribution on  $\Omega = \prod_{i=1}^n \Omega'_i$ , and consider the down and up operators defined with respect to its homogenization  $\mu^{\text{hom}}$ .*

1. *If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in  $\chi^2$ , then  $D_{k \rightarrow \ell}$  satisfies  $\prod_{j=\ell+1}^k \left( 1 - \frac{1}{j(C + \frac{n-j+1}{n})} \right)$ -contraction in  $\chi^2$ .*
2. *If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in KL (i.e.,  $\mu$  satisfies  $C$ -approximate tensorization of entropy), then  $D_{k \rightarrow \ell}$  and  $P_{k \rightarrow \ell}^\nabla$  satisfy  $\prod_{j=\ell+1}^k \left( 1 - \frac{1}{j(C+1)} \right)$ -contraction in KL.*

We note that our tensorization construction in Lemma 3.1 currently depends on  $\mu$  being a homogeneous distribution; it would be interesting to extend it beyond this case.

**Proof.** If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in  $\chi^2$ -divergence, then by Lemma 3.4,  $D_{j \rightarrow j-1}$  satisfies  $\left( 1 - \frac{1}{j(C + \frac{n-j+1}{n})} \right)$ -contraction in  $\chi^2$ -divergence. If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in  $\chi^2$ -divergence, then we have  $\left( 1 - \frac{1}{j(C+1)} \right)$ -contraction in KL-divergence. Because  $D_{k \rightarrow \ell} = D_{k \rightarrow k-1} \cdots D_{\ell+1 \rightarrow \ell}$ , taking the product from  $\ell + 1$  to  $k$  gives the result.  $\blacktriangleleft$

From this, we can conclude that  $k$ -Glauber dynamics mixes at least  $\Omega(k)$  times as fast in  $\chi^2$ -divergence as Glauber dynamics, and assuming approximate tensorization of entropy, mixes at least  $\Omega(k)$  times as fast in KL-divergence.

**► Corollary 3.6** ( $k$ -Glauber mixes  $k$  times as fast). *Let  $\mu$  be a distribution on  $\Omega = \prod_{i=1}^n \Omega'_i$ .*

1. *If  $\mu$  satisfies a Poincaré inequality with constant  $Cn$ , then*

$$\lambda_2(P_{\mu,k}) \leq \left( 1 - \frac{k}{n+1} \right)^{\frac{1}{C+1}} = 1 - \Omega \left( \max \left\{ \frac{k}{(C+1)n}, 1 \right\} \right)$$

*and  $P_{\mu,k}$  satisfies a Poincaré inequality with constant  $\Omega \left( \frac{(C+1)n}{k} \right)$ .*

2. *If  $\mu$  satisfies  $C$ -approximate tensorization of entropy, then  $P_{\mu,k}$  satisfies contraction in KL-divergence with constant*

$$\left( 1 - \frac{k}{n+1} \right)^{\frac{1}{C+1}} = 1 - \Omega \left( \max \left\{ \frac{k}{(C+1)n}, 1 \right\} \right).$$

**Proof.** Note that  $P_{\mu,k}$  corresponds to  $P_{n \leftrightarrow n-k}^\nabla$  after homogenization. For variance, we recall Proposition 2.8 which relates the Poincaré constant of  $P_\mu$  and  $D_{n \rightarrow n-1}$ , and the Poincaré constant of  $P_{\mu,k}$  and  $D_{n \rightarrow n-k}$ . The corollary then follows from Theorem 3.5 and the calculation

$$\begin{aligned} \prod_{j=n-k+1}^n \left(1 - \frac{1}{j(C+1)}\right) &\leq e^{-\frac{1}{C+1} \sum_{j=n-k+1}^n \frac{1}{j}} \leq e^{-\frac{1}{C+1} \ln\left(\frac{n+1}{n-k+1}\right)} \\ &= \left(1 - \frac{k}{n+1}\right)^{\frac{1}{C+1}} = 1 - \Omega\left(\frac{k}{(C+1)n}\right). \quad \blacktriangleleft \end{aligned}$$

Our theorem also implies contraction for  $D_{2 \rightarrow 1}$ ; this forms a kind of converse to local-to-global arguments that start with contraction of  $D_{2 \rightarrow 1}$  [2].

► **Corollary 3.7.** *Let  $\mu$  be a distribution on  $\Omega = \prod_{i=1}^n \Omega'_i$ , and consider the down and up operators defined with respect to its homogenization  $\mu^{\text{hom}}$ .*

1. *If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in  $\chi^2$ , then  $D_{2 \rightarrow 1}$  satisfies  $(1 - \frac{1}{2(C+1)})$ -contraction in  $\chi^2$  and  $\lambda_2(P_{1 \leftrightarrow 2}^\Delta) \leq 1 - \frac{1}{2(C+1)}$ .*
2. *If  $D_{n \rightarrow n-1}$  satisfies  $(1 - \frac{1}{Cn})$ -contraction in KL (i.e.,  $\mu$  satisfies  $C$ -approximate tensorization of entropy), then  $D_{2 \rightarrow 1}$  satisfies  $(1 - \frac{1}{2(C+1)})$ -contraction in KL.*

**Proof.** This follows from substituting  $k = 2$ ,  $\ell = 1$  into Theorem 3.5, appealing to Proposition 2.8 for the  $\chi^2$  result. ◀

## 4 Parallel sampling for Ising models

To apply Corollary 3.6 to the Ising model for  $\|J\| < 1$ , we use the fact that the Ising model satisfies approximate tensorization of entropy.

► **Theorem 4.1.** *Suppose  $\|J\| < 1$ . Then  $\mu_{J,h}$  satisfies approximate tensorization of entropy with constant  $\frac{1}{1-\|J\|}$ .*

The proof is in Section 4.1. We first introduce a generic guarantee for approximate rejection sampling in Section 4.2, based on establishing concentration for the difference of the log-pdfs. In Section 4.3, we show that using an approximating product distribution – chosen as the solution to a variational problem – is sufficient as a proposal distribution for  $\mu_{J,h}$  when  $\|J\|_F$  is small enough, using the Hanson-Wright inequality. We put everything together in Section 4.4, combining the speedup of  $k$ -Glauber dynamics, known mixing for the Ising model, guarantee on the approximate rejection sampler, with a careful analysis of the recursion in the algorithm.

### 4.1 Approximate tensorization of entropy for the Ising model

For  $\|J\| < 1$ , we prove approximate tensorization of entropy for the Ising model  $\mu_{J,h}$  (Theorem 4.1) by using the fact that it holds for rank-1 Ising models and using the needle decomposition in [24]. This is the same way that the modified log-Sobolev inequality is proved in [24].

► **Proposition 4.2** ([8, Proposition 32]). *Suppose  $\|u\| < 1$ . Then  $\mu_{uu^\top, h}$  satisfies approximate tensorization of entropy with constant  $\frac{1}{1-\|u\|^2}$ .*

► **Theorem 4.3** (Needle decomposition of Ising measures [24]). *Consider an Ising model  $\mu = \mu_{J,h}$  on  $\{\pm 1\}^n$  with  $J \geq 0$ . Let  $f : \{\pm 1\}^n \rightarrow \mathbb{R}$  be any function. There exists a mixture decomposition (depending on  $f$ )*

$$\mu(x) = \int \mu_{u,v}(x) d\pi(u, v)$$

where  $\pi$  is a probability measure on  $\mathbb{R}^{2n}$  such that:

1.  $\pi$ -almost surely,  $\mu_{u,v}$  is a probability measure of the form

$$\mu_{u,v}(x) = \frac{1}{Z_{u,v}} \exp\left(\frac{1}{2} \langle u, x \rangle^2 + \langle v, x \rangle\right),$$

i.e., a rank one Ising model (“needle”). Furthermore  $\|u\| \leq \|J\|$ .

2.  $\mathbb{E}_{\mu_{u,v}} f(X) = \mathbb{E}_{\mu} f(X)$   $\pi$ -almost surely.

**Proof of Theorem 4.1.** See full version. ◀

## 4.2 Approximate rejection sampling

Conditional distributions of the Ising model are again Ising models. We will show that with large probability, if we pick a random subset of not-too-large size, then we can approximate the distribution of those coordinates with a product distribution, and hence use the product distribution as a proposal for approximate rejection sampling. We discuss further the choice of product distribution (given in Algorithm 2) in Section 4.3.

First, we give a generic guarantee for the rejection sampling Algorithm 3, which can be used whenever the log-ratio between desired and proposal distributions  $\ln \frac{dP}{dQ}$  has sufficiently decaying exponential tails. Note that because the normalizing constants are unknown, we draw two samples and use one as a reference to decide whether to accept the other. Lemma 4.4 appears as [25, Lemma 2] specialized to the distribution they consider, but holds more generally. We give the proof for completeness.

■ **Algorithm 2** Quadratic approximate rejection sampler (QuadraticApproxRejectionSampler).

- 
- 1: **Input:** Hamiltonian  $H$  in the form  $H(x) = C + \langle h, x \rangle + \frac{1}{2} \langle x, Ax \rangle + H_{\geq 3}(x)$ ,  $\delta$  such that  $\|A^{\otimes}\|_F \leq \delta < c_3$  and  $0 \preceq A \preceq (1-c)I$ , error  $\varepsilon$ .
  - 2: Let  $u_0 = \mathbf{0}$ .
  - 3: **for**  $t$  from 1 to  $T = \Theta\left(\frac{1}{c} \ln\left(\frac{\sqrt{n}}{\delta}\right)\right)$  **do**
  - 4:     Let  $u_t = A^{\otimes} \tanh(h + u_{t-1})$ .
  - 5: **end for**
  - 6: Let  $\hat{h} = u_T$ .
  - 7: **Output:**  $\text{ApproxRejectionSampler}(q(x) \propto \exp(\langle h + \hat{h}, x \rangle), g(x) = H(x) - \langle h + \hat{h}, x \rangle, (\frac{2}{\varepsilon})^{\frac{\delta}{c_3 - \delta}})$   
(See Algorithm 3.)
- 

■ **Algorithm 3** Approximate rejection sampler (ApproxRejectionSampler).

- 
- 1: **Input:** Oracle for sampling from  $Q$ , function  $g$  such that  $\frac{dP}{dQ} \propto e^g$ , error parameter  $c$ .
  - 2: **repeat** (For parallel implementation, run  $\lceil c \rceil$  times simultaneously and take the first success.)
  - 3:     Draw  $X, Z \sim Q$ .
  - 4:     Let  $R = \exp(g(X) - g(Z))$ .
  - 5:     Draw  $U \sim \text{Uniform}([0, 1])$ .
  - 6: **until**  $U \leq \frac{1}{c}R$
  - 7: **Output:**  $X$ .
- 

► **Lemma 4.4.** Let  $\hat{P}$  be the distribution of the output of Algorithm 3. Then

$$\frac{dP}{dQ}(X) = \frac{\mathbb{E}[R|X]}{\mathbb{E}R} \qquad \frac{d\hat{P}}{dQ}(X) = \frac{\mathbb{E}[\min\{R, c\}|X]}{\mathbb{E}[\min\{R, c\}]}$$

## 49:16 Parallelising Glauber Dynamics

The acceptance probability is  $p_{\text{accept}} = \frac{1}{c} \mathbb{E} \min\{R, c\} = \frac{1}{c} (\mathbb{E}R - \mathbb{E}[(R - c)\mathbb{1}_{R \geq c}])$  and the TV distance is bounded by

$$\mathcal{D}_{\text{TV}}(\widehat{P}, P) \leq \frac{\mathbb{E}[(R - c)\mathbb{1}_{R \geq c}]}{\mathbb{E}R}.$$

For  $c \geq 1$ , the acceptance probability is at least  $\frac{1}{2c}$ .

**Proof.** We calculate

$$\begin{aligned} \mathbb{E}[R|X] &= \mathbb{E}[e^{g(X)-g(Z)}|X] = e^{g(X)} \mathbb{E}[e^{-g(Z)}] \\ \mathbb{E}[R] &= \mathbb{E}[\mathbb{E}[R|X]] = \mathbb{E}[e^{g(X)}] \mathbb{E}[e^{-g(Z)}], \end{aligned}$$

so  $\frac{\mathbb{E}[R|X]}{\mathbb{E}[R]} = \frac{e^{g(X)}}{\mathbb{E}[e^{g(X)}]} = \frac{d\widehat{P}}{dQ}(X)$ . Note  $\frac{d\widehat{P}}{dQ}(X)$  is the probability of acceptance given  $X$  divided by the total probability of acceptance, which we calculate:

$$\begin{aligned} p_{\text{accept}}(X) &:= \mathbb{P}\left[U \leq \frac{1}{c}R|X\right] = \mathbb{E}\left[\min\left\{\frac{R}{c}, 1\right\}|X\right] = \frac{1}{c} \mathbb{E}[\min\{R, c\}|X] \\ p_{\text{accept}} &= \mathbb{P}\left[U \leq \frac{1}{c}R\right] = \mathbb{E}\left[\mathbb{P}\left[U \leq \frac{1}{c}R|X\right]\right] = \frac{1}{c} \mathbb{E}[\mathbb{E}[\min\{R, c\}|X]] = \frac{1}{c} \mathbb{E}[\min\{R, c\}]. \end{aligned}$$

Dividing gives  $\frac{d\widehat{P}}{dQ} = \frac{\mathbb{E}[\min\{R, c\}|X]}{\mathbb{E}[\min\{R, c\}]}$ . Then

$$\begin{aligned} \mathcal{D}_{\text{TV}}(\widehat{P}, P) &\leq \mathbb{E}_{X \sim Q} \max\left\{0, \frac{dP}{dQ}(X) - \frac{d\widehat{P}}{dQ}(X)\right\} \\ &\leq \mathbb{E}_{X \sim Q} \max\left\{0, \frac{\mathbb{E}[R|X]}{\mathbb{E}R} - \frac{\mathbb{E}[\min\{R, c\}|X]}{cp_{\text{accept}}}\right\} \\ &\leq \mathbb{E}_{X \sim Q} \max\left\{0, \frac{\mathbb{E}[R|X]}{\mathbb{E}R} - \frac{\mathbb{E}[\min\{R, c\}|X]}{\mathbb{E}R}\right\} && \text{because } cp_{\text{accept}} \leq \mathbb{E}R \\ &\leq \mathbb{E}_{X \sim Q} \frac{\mathbb{E}[(R - c)\mathbb{1}_{R \geq c}|X]}{\mathbb{E}R} = \frac{\mathbb{E}[(R - c)\mathbb{1}_{R \geq c}]}{\mathbb{E}R}. \end{aligned}$$

Finally, note that  $\mathbb{P}(R \geq 1) \geq \frac{1}{2}$  by symmetry, so  $p_{\text{accept}} \geq \frac{1}{c} \mathbb{P}(R \geq 1) \geq \frac{1}{2c}$ .  $\blacktriangleleft$

### 4.3 Concentration

To obtain concentration of the ratio in Lemma 4.4, we need a version of the Hanson-Wright inequality. We first state the classical inequality.

► **Theorem 4.5** (Hanson-Wright Inequality, [49, Theorem 6.2.1]). *There is a constant  $c$  such that the following holds. Let  $X = (X_1, \dots, X_n) \in \mathbb{R}^n$  be a random vector with independent, mean-zero,  $K$ -sub-gaussian coordinates. Let  $A \in \mathbb{R}^{n \times n}$  be a matrix. Then for every  $t \geq 0$ ,*

$$\mathbb{P}(|\langle X, AX \rangle - \mathbb{E}\langle X, AX \rangle| \geq t) \leq 2 \exp\left[-c \min\left\{\frac{t^2}{K^4 \|A\|_F^2}, \frac{t}{K^2 \|A\|}\right\}\right].$$

We will use the following version, which is a consequence of [44, Corollary 2] (by taking  $\Gamma(f) = \|\nabla f\|$  and noting that product distributions on  $\{\pm 1\}^n$  satisfy a uniform modified log-Sobolev inequality) and allows a general function  $f$ .

► **Theorem 4.6** ([44]). *There is a constant  $c$  such that the following holds. Let  $X = (X_1, \dots, X_n) \in \{\pm 1\}^n$  be a random vector with independent coordinates. Let  $f : \{\pm 1\}^n \rightarrow \mathbb{R}$  be a function. Then for every  $t \geq 0$ ,*

$$\mathbb{P}(|f(X) - \mathbb{E}f(X)| \geq t) \leq 2 \exp\left[-c \min\left\{\frac{t^2}{\mathbb{E}[\|\nabla f\|^2]}, \frac{t}{\max_{x \in \{\pm 1\}^n} \|\nabla^2 f\|_F}\right\}\right].$$



Note the extra term  $\mathbb{E}[\|\nabla f\|^2]$  compared to Theorem 4.5 which requires the random variables to be centered. This means that we cannot simply take  $Q = \mu_h := \mu_{O,h}$  and  $P = \mu_{J,h}$  for the reason that  $\mathbb{E}[\|\nabla_x(\langle x, Ax \rangle)\|^2]$  can be  $\Omega(n)$ , while we need concentration to  $O(1)$ . Instead, in order to apply Theorem 4.6 for  $f = \ln \frac{dP}{dQ}$ , we would like to  $\nabla f$  to be centered, that is,  $\mathbb{E}_Q \nabla f = 0$ . For this, we need to solve the variational problem

$$\mathbb{E}_{\mu_{h+h^*}} A^{\otimes} x = h^*. \quad (3)$$

We do this by fixed point iteration. Note this is a special case of “gradient” descent for Lipschitz and strongly monotone operators [15, 36].

► **Lemma 4.7** (Fixed point iteration). *Let  $(X, d)$  be a metric space,  $c > 0$ , and suppose  $F : X \rightarrow X$  is  $(1-c)$ -Lipschitz (so it is a contraction mapping). Let  $x_0 \in X$  and  $x_t = F^{(t)}(x_0)$ . Then*

$$d(F(x_t), x_t) \leq (1-c)^t d(F(x_0), x_0)$$

and hence for  $t = \Omega\left(\frac{1}{c} \ln\left(\frac{d(F(x_0), x_0)}{\varepsilon}\right)\right)$ , we have  $d(F(x_t), x_t) \leq \varepsilon$ .

**Proof.** We have  $d(F(x_t), x_t) = d(F^{(t+1)}(x_0), F^{(t)}(x_0)) \leq (1-c)^t d(F(x_0), x_0)$  by induction. Hence, it suffices to choose  $t$  such that  $t \ln\left(\frac{1}{1-c}\right) \geq \ln\left(\frac{d(F(x_0), x_0)}{\varepsilon}\right)$ , which gives the result. ◀

► **Lemma 4.8.** *Suppose that  $A$  is symmetric positive semi-definite with  $A \preceq (1-c)I$ . Let*

$$F(u) = A^{\otimes} \tanh(h + u).$$

Then for  $t = \Omega\left(\frac{1}{c} \log\left(\frac{\sqrt{n}}{\varepsilon}\right)\right)$ , we have that  $\hat{h} := F^{(t)}(\mathbf{0})$  satisfies

$$\left\| \mathbb{E}_{\mu_{h+\hat{h}}} A^{\otimes} x - \hat{h} \right\| \leq \varepsilon. \quad (4)$$

**Proof.** Note that all diagonal entries of  $A$  are contained in  $[0, 1-c]$ , so  $-(1-c)I \preceq A^{\otimes} \preceq (1-c)I$ . Combining this with the fact that  $\tanh$  is 1-Lipschitz, we obtain that  $F$  is  $(1-c)$ -Lipschitz. Note that  $\|F(\mathbf{0}) - \mathbf{0}\| \leq \sqrt{n}$ . The result then follows from Lemma 4.7 and the fact that  $F(u) = \mathbb{E}_{\mu_{h+u}} A^{\otimes} x$ . ◀

Using this, for small enough  $\|A\|_F$ , we can obtain the exponential tails necessary to bound the TV-distance in Lemma 4.4. This fits in with the general fact that Ising models with small  $\|J\|_F$  are well-approximated by product distributions [31], giving approximation guarantees for variational methods in this regime. We state the following lemma more generally with a higher-order term, so that we can also apply it for the  $p$ -spin model.

► **Lemma 4.9.** *There is a constant  $c_3$  such that the following holds. Suppose  $H(x) = \langle h, x \rangle + \frac{1}{2} \langle x, Ax \rangle + H_{\geq 3}(x)$  where  $A$  is symmetric and  $H_{\geq 3}(x) = \sum_{|I| \geq 3} a_I x^I$  contains the terms of degree  $\geq 3$ . If  $\delta < c_3$ ,*

$$\max_{x \in \{\pm 1\}^n} \|\nabla^2 H(x)\|_F \leq \delta \quad \text{and} \quad \max_{x \in \{\pm 1\}^n} \|\nabla H_{\geq 3}(x)\| \leq \delta,$$

then the output of `QuadraticApproxRejectionSampler` (Algorithm 2) is at most  $\varepsilon$  in TV distance from  $\mu$ , and the acceptance probability in the call to `ApproxRejectionSampler` is at least  $\frac{1}{2c} = \frac{1}{2} \left(\frac{\varepsilon}{2}\right)^{\frac{\delta}{c_3 - \delta}}$ .

In the special case that  $H_{\geq 3}(x) = 0$ , the assumption simplifies to  $\|A^{\otimes}\|_F \leq \delta$ .

**Proof.** Let

$$f(x) = H(x) - \langle h + \hat{h}, x \rangle = \frac{1}{2} \langle x, Ax \rangle - \langle \hat{h}, x \rangle + H_{\geq 3}(x).$$

To use Theorem 4.6, we calculate  $\mathbb{E}[\|\nabla f\|^2]$ . First note that  $\mathbb{E}_{x \sim \text{Uniform}(\{\pm 1\}^n)} \nabla^2 H_{\geq 3}(x) = O$  (because for any  $i, j \in [n]$  and  $|I| \geq 3$ , we have  $\mathbb{E}_{x \sim \text{Uniform}(\{\pm 1\}^n)} x^I \setminus \{i, j\} = 0$ ) so by Jensen's inequality

$$\begin{aligned} \left\| A^{\otimes} \right\|_F &= \left\| A^{\otimes} + \mathbb{E}_{x \sim \text{Uniform}(\{\pm 1\}^n)} \nabla^2 H_{\geq 3}(x) \right\|_F \\ &\leq \mathbb{E}_{x \sim \text{Uniform}(\{\pm 1\}^n)} \left\| A^{\otimes} + \nabla^2 H_{\geq 3}(x) \right\|_F \leq \max_{x \in \{\pm 1\}^n} \left\| \nabla^2 H(x) \right\|_F \leq \delta. \end{aligned}$$

Let  $\bar{x} = \mathbb{E}_{\mu_{h+\hat{h}}} x$ . From Lemma 4.8 we have that the output  $\hat{h}$  of fixed point iteration satisfies  $\left\| A^{\otimes} \bar{x} - \hat{h} \right\| \leq \delta$ . We then have

$$\begin{aligned} \mathbb{E}_{\mu_{h+\hat{h}}} [\|\nabla f\|^2] &= \mathbb{E}_{\mu_{h+\hat{h}}} \left[ \left\| A^{\otimes} x - \hat{h} + \nabla H_{\geq 3}(x) \right\|^2 \right] \\ &\leq 2\mathbb{E}_{\mu_{h+\hat{h}}} \left[ \left\| A^{\otimes} (x - \bar{x}) + A^{\otimes} \bar{x} - \hat{h} \right\|^2 \right] + 2\mathbb{E}_{\mu_{h+\hat{h}}} \left[ \|\nabla H_{\geq 3}(x)\|^2 \right] \\ &= 2\mathbb{E}_{\mu_{h+\hat{h}}} \left[ \left\| A^{\otimes} \bar{x} - \hat{h} \right\|^2 + \left\| A^{\otimes} (x - \bar{x}) \right\|^2 + \|\nabla H_{\geq 3}(x)\|^2 \right] \\ &\leq 2 \left[ \left\| A^{\otimes} \bar{x} - \hat{h} \right\|^2 + \left\| A^{\otimes} \right\|_F^2 + \mathbb{E}_{\mu_{h+\hat{h}}} \|\nabla H_{\geq 3}(x)\|^2 \right] \leq 6\delta^2 \end{aligned} \quad (5)$$

using the fact that the entries of  $x - \bar{x}$  are independent, mean 0, with variance at most 1. Hence, by Theorem 4.6,  $f(X) - \mathbb{E}f(X)$  is  $O(\delta)$ -sub-exponential, and so is  $f(Z) - f(X)$ , and there exists  $c_3$  so that

$$\mathbb{P}(|f(Z) - f(X)| \geq t) \leq 2e^{-\frac{c_3 t}{\delta}}.$$

Then for  $c = \left(\frac{2}{\varepsilon}\right)^{\frac{\delta}{c_3 - \delta}}$ ,

$$\begin{aligned} \mathbb{E}[(R - c)\mathbb{1}_{R \geq c}] &\leq \int_{\ln c}^{\infty} e^t \cdot \mathbb{P}(f(Z) - f(X) \geq t) dt \\ &\leq \int_{\ln c}^{\infty} e^t 2e^{-\frac{c_3 t}{\delta}} dt \leq 2 \int_{\ln c}^{\infty} e^{-(\frac{c_3}{\delta} - 1)t} dt = 2c^{-(\frac{c_3}{\delta} - 1)} = \varepsilon. \end{aligned} \quad (6)$$

Moreover, by Jensen's inequality,  $\mathbb{E}R \geq e^{\mathbb{E}[f(Z) - f(X)]} = 1$ . Hence by Theorem 4.6, the output is at most  $\varepsilon$  in TV distance from  $\mu$  and the acceptance probability is at least  $\frac{1}{2c}$ . ◀

#### 4.4 Analysis of the Parallel Ising Sampler

► **Lemma 4.10.** *Let  $S \subseteq [n]$ , fix  $x_{S^c} \in \{\pm 1\}^{S^c}$ , and let  $P$  be the distribution on  $\{\pm 1\}^S$  with mass function  $p(x) = \mu_{J,h}(X_S = x | X_{S^c} = x_{S^c})$ , and let  $Q$  be the product distribution in on  $\{\pm 1\}^S$  with mass function  $q(x) \propto \exp(\langle J_{S \times S^c} x_{S^c} + h_S, x \rangle)$ . Then the following hold.*

1.  $\frac{dP}{dQ}(x) \propto \exp\left(\frac{1}{2} \langle x, J_{S \times S} x \rangle\right)$ .
2.  $\mathcal{D}_{\text{KL}}(P \| Q) \leq \|J_{S \times S}\| \cdot |S|$ .

**Proof.** Because  $x_{S^c}$  is constant, expanding the quadratic gives

$$\mu_{J,h}(X_S = x_S | X_{S^c} = x_{S^c}) \propto \exp\left(\frac{1}{2} (2 \langle x_S, J_{S \times S^c} x_{S^c} \rangle + \langle x_S, J_{S \times S} x_S \rangle) + \langle h_S, x_S \rangle\right).$$

Dividing by  $q(x_S)$  gives (1).

For (2), we note that for  $x \in \{\pm 1\}^S$ ,  $|\frac{1}{2} \langle x, J_{S \times S} x \rangle| \leq \frac{1}{2} \|J_{S \times S}\| \|x\|^2 \leq \frac{1}{2} \|J_{S \times S}\| |S|$ . Hence

$$\frac{dP}{dQ}(x) = \frac{\exp\left(\frac{1}{2} \langle x, J_{S \times S} x \rangle\right)}{\int \exp\left(\frac{1}{2} \langle x, J_{S \times S} x \rangle\right) dQ(x)} \leq \frac{e^{\frac{1}{2} \|J_{S \times S}\| |S|}}{e^{-\frac{1}{2} \|J_{S \times S}\| |S|}} = e^{\|J_{S \times S}\| |S|}$$

and  $\mathcal{D}_{\text{KL}}(P \| Q) = \mathbb{E}_P \ln \frac{dP}{dQ} \leq \|J_{S \times S}\| |S|$ .  $\blacktriangleleft$

► **Lemma 4.11** (Bernstein's inequality for supermartingales [28, (1.6)]). *Let  $X_n$  be a martingale adapted to  $\mathcal{F}_n$ . Suppose that  $|X_{n+1} - X_n| \leq L$  and  $\mathbb{E}[|X_{n+1} - X_n|^2 | \mathcal{F}_n] \leq \sigma^2$  with probability 1. Then*

$$\mathbb{P}(X_n - X_0 \geq t) \leq \exp\left(-\frac{t^2}{2(tL + n\sigma^2)}\right).$$

We are now ready to prove our main theorem on the parallel Ising sampler.

**Proof of Theorem 1.2.** We first note that all lines in Algorithm 1 take logarithmic time with  $\text{poly}(n)$  processors (e.g., by a parallel implementation of matrix-vector multiplication). Note that a random subset of specified size  $s$  can be selected by generating a random number for each index, using a parallel sorting algorithm [11], and then selecting the smallest  $s$  elements. We will ignore logarithmic overhead for the rest of the proof.

**Running time is bounded with high probability.** We consider a tree associated with a run of the algorithm, where each node is labeled with a set, constructed as follows. Each node represents a time that `ParallelsingSampler` is called, and each leaf node represents a time that `ApproxRejectionSampler`. Start with a root node  $v_1$  labeled with  $S_1 = [n]$ . A node has  $T$  children, where  $T$  is the number calculated in line 7 of the algorithm. Each node is labeled with subset of indices marking out the submatrix  $J_{S \times S}$  it is given.

Now consider exploring the tree in the following breadth-first manner. We will define a list  $B_t$  which will contain the vertices at the boundary of explored territory and a filtration  $\mathcal{F}_t$ . Let  $B_0 = (v_1)$  and  $\mathcal{F}_0$  be the trivial  $\sigma$ -algebra. Given  $B_t$  and  $\mathcal{F}_t$ , if  $B_t$  is non-empty, define  $B_{t+1}$  and  $\mathcal{F}_{t+1}$  as follows. Let  $v_{t+1}$  be the first vertex in the list  $B_t$ , and let  $B_{t+1}$  be defined from  $B_t$  by removing  $v_{t+1}$  from  $B_t$  and adding its children. Let  $S_{t+1}$  denote the set of indices associated with  $v_{t+1}$ , considered as a set-valued random variable, and  $\mathcal{F}_{t+1} = \sigma(\mathcal{F}_t, S_{t+1})$ . Let  $M_t = |B_t|$ . We have  $M_1 = \lfloor C_2 \ln\left(\frac{n}{\varepsilon}\right) \frac{n}{s} \rfloor$ , and wish to bound the first time  $\tau$  such that  $M_\tau = 0$ . We redefine  $M_{\tau+k} = -k$  (for sake of making  $M_t$  a supermartingale, as we will show below).

For  $t \geq 2$ , consider  $M_t - M_{t-1} | \mathcal{F}_{t-1}$ . Let  $v$  denote the parent of  $v_t$ , and suppose  $v$  is

associated with the set  $R$ , with  $|R| = m$ . Then  $|S_t| = s := \left\lceil \frac{c_1 m}{\left(\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right) + 1\right) \ln\left(\frac{n}{\varepsilon}\right)} \left\| J_{R \times R}^\otimes \right\|_F \right\rceil$ .

(We choose  $c_1 \leq \frac{1}{2} c_3$  to ensure that we always have  $s \leq m$ .) Let  $D_t$  be the number of new children added. If  $\left\| J_{S_t \times S_t}^\otimes \right\|_F \leq c_3$  or  $s = 1$ , then  $v_t$  is a leaf and  $D_t = 0$ . Now consider

$\left\| J_{S_t \times S_t}^\otimes \right\|_F > c_3$ . In the current call to the algorithm,  $s' = \left\lceil \frac{c_1 s}{\left(\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right) + 1\right) \ln\left(\frac{n}{\varepsilon}\right)} \left\| J_{S_t \times S_t}^\otimes \right\|_F \right\rceil$ .

Then  $D_t \leq C_2 \ln\left(\frac{n}{\varepsilon}\right) \frac{s}{s'} \leq \frac{C_2 \left(\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right) + 1\right) \ln\left(\frac{n}{\varepsilon}\right)^2 \left\| J_{S_t \times S_t}^\otimes \right\|_F}{c_1}$ . In either case,  $M_t - M_{t-1} = D_t - 1$ . We have that

$$D_t \leq \frac{C_2 \left(\ln\left(\frac{2}{\varepsilon_{\text{step}}}\right) + 1\right) \ln\left(\frac{n}{\varepsilon}\right)^2 \left\| J_{S_t \times S_t}^\otimes \right\|_F \mathbb{1} \left[ \left\| J_{S_t \times S_t}^\otimes \right\|_F > c_3 \right]}{c_1}. \quad (7)$$

Hence, by Cauchy-Schwarz and Chebyshev's inequality,

$$\begin{aligned}
\mathbb{E}[D_t | \mathcal{F}_{t-1}] &\leq \frac{C_2 \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right) \ln \left( \frac{n}{\varepsilon} \right)^2}{c_1} \mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right]^{1/2} \\
&\quad \mathbb{P} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F > \frac{c_3}{\ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1} | \mathcal{F}_{t-1} \right]^{1/2} \\
&\leq \frac{C_2 \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right) \ln \left( \frac{n}{\varepsilon} \right)^2}{c_1} \mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right]^{1/2} \cdot \frac{\mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right]^{1/2}}{c_3 / \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right)} \\
&= \frac{C_2 \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right)^2 \ln \left( \frac{n}{\varepsilon} \right)^2}{c_1 c_3} \mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right]. \tag{8}
\end{aligned}$$

Now because  $S_t$  is uniformly chosen at random from subsets of  $R$  of size  $s$ ,

$$\begin{aligned}
\mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right] &= \mathbb{E}_{S \sim \text{Uniform} \binom{R}{s}} \left[ \left\| J_{S \times S}^{\otimes} \right\|_F^2 \right] \\
&= \sum_{i,j \in R, i \neq j} \binom{s}{m} J_{ij}^2 = \binom{s}{m}^2 \left\| J_{R \times R}^{\otimes} \right\|_F^2 \leq \frac{4c_1^2}{\left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right)^2 \ln \left( \frac{n}{\varepsilon} \right)^2}, \tag{9}
\end{aligned}$$

where we use the fact that  $J_{ii}^{\otimes} = 0$ , all off-diagonal entries have probability  $\left(\frac{s}{m}\right)^2$  of being included in  $S_t$ , and  $s > 1$ . Combining (8) and (9) gives

$$\mathbb{E}[D_t | \mathcal{F}_{t-1}] \leq \frac{4C_2 c_1}{c_3}.$$

Choosing  $c_1$  small enough (depending on  $C_2, c_3$ ), we can ensure that  $\mathbb{E}[M_t - M_{t-1} | \mathcal{F}_{t-1}] = \mathbb{E}[D_t - 1 | \mathcal{F}_{t-1}] \leq -\frac{1}{2}$ , so that  $M_t + \frac{t}{2}$  is a supermartingale for  $t \geq 1$ . By Doob's decomposition we can write  $M_t = A_t + M'_t$  where  $A_{t+1} \leq A_t - \frac{t}{2}$  is a predictable decreasing sequence and  $M'_t$  is a martingale.

We now bound the variance. Using (7),

$$\begin{aligned}
\mathbb{E}[(M'_t - M'_{t-1})^2 | \mathcal{F}_{t-1}] &\leq \mathbb{E}[D_t^2 | \mathcal{F}_{t-1}] \\
&\leq \frac{C_2^2 \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right)^2 \ln \left( \frac{n}{\varepsilon} \right)^4}{c_1^2} \mathbb{E} \left[ \left\| J_{S_t \times S_t}^{\otimes} \right\|_F^2 | \mathcal{F}_{t-1} \right] \\
&\leq 4C_2^2 \ln \left( \frac{n}{\varepsilon} \right)^2,
\end{aligned}$$

where we use the bound (9). Finally,  $|M'_{t+1} - M'_t| \leq \frac{C_2 \left( \ln \left( \frac{2}{\varepsilon_{\text{step}}} \right) + 1 \right) \ln \left( \frac{n}{\varepsilon} \right)^2}{c_1} \left\| J^{\otimes} \right\|_F$  with probability 1. Let  $T_0$  be the  $T$  computed in line 7 in the first step of the algorithm. By Bernstein's inequality for martingales (Lemma 4.11), for  $t \geq C \ln^4 \left( \frac{n}{\varepsilon} \right) \max \left\{ \left\| J^{\otimes} \right\|_F, 1 \right\} \geq T_0$  for an appropriate constant  $C$  (depending on  $C_2, c_1, C_4$ ),

$$\mathbb{P}(M_{t+1} > 0) = \mathbb{P}((M_{t+1} - M_1) > -T_0) \leq \mathbb{P} \left( M'_{t+1} - M'_1 > \frac{t}{2} - T_0 \right) \leq \frac{\varepsilon}{2}.$$

This shows that with probability  $\geq 1 - \frac{\varepsilon}{4}$ , there are at most  $t_{\max} = C \ln^4 \left( \frac{n}{\varepsilon} \right) \max \{ \|J\|_F, 1 \}$  nodes.

Finally, we note that in the call to `ApproxRejectionSampler`, the parameter needed to obtain error  $\varepsilon_{\text{step}}$  is

$$c = \left( \frac{2}{\varepsilon_{\text{step}}} \right)^{\frac{\delta}{c_3 - \delta}} = \left( \frac{2}{\varepsilon_{\text{step}}} \right)^{\frac{1}{\log(2/\varepsilon_{\text{step}})}} = e$$

and the acceptance probability is  $\geq \frac{1}{2c} = \frac{1}{2e}$ .

The number of tries until acceptance is a geometric random variable, which is subexponential, so standard concentration bounds show that the total number of tries is at most  $O(\ln(\frac{1}{\varepsilon}))$  times the number of calls, with probability  $\geq 1 - \frac{\varepsilon}{2}$ . Putting everything together, we obtain  $O(\max\{\|J\|_F, 1\} \text{poly log}(\frac{n}{\varepsilon}))$  running time with probability  $\geq 1 - \varepsilon$ .

**Output is close in TV distance.** Let  $A$  be a large constant to be determined.

Now consider coupling  $y = y^{(0)}$  with a sequence of random variables  $y^{(1)}, \dots$ , defined inductively as follows. Start with all vertices of the tree of recursive calls unmarked. Now given  $y^{(t)}$ , choose a node (in a fixed manner) all of whose children are marked, and mark it; then replace the output of that call to `ParallellisingSampler` by a sample from the true distribution. We now choose constants so that  $\mathcal{D}_{\text{TV}}(\mathcal{D}(y^{(t)}), \mathcal{D}(y^{(t+1)})) \leq \frac{\varepsilon}{n^A}$ . There are two kinds of replacements to consider, a leaf node and a non-leaf node.

A leaf node corresponds to a call to `ApproxRejectionSampler`. If  $c_3$  is small enough and  $C_4 = A$ , then by Lemma 4.9 and 4.4, the output of `ApproxRejectionSampler` is within  $\frac{\varepsilon}{n^A}$  of the  $\mu_{J_{R \times R}, h}$ .

A non-leaf node corresponds to  $T$  recursive calls to `ParallellisingSampler`. Here we must appeal to mixing for the Ising model. By Theorem 4.1, approximate tensorization of entropy holds with constant  $\frac{1}{c}$ . Hence by Theorem 3.6, there is a constant  $C'_0$  such that if  $C_0 = C'_0 c$ , then for any  $s, t \cdot \frac{n}{s}$  steps of  $s$ -Glauber dynamics results in a distribution  $\nu_t$  satisfying

$$\mathcal{D}_{\text{TV}}(\nu_t \| \mu_{J, h}) \leq \sqrt{\frac{1}{2} \mathcal{D}_{\text{KL}}(\nu_t \| \mu_{J, h})} \leq \sqrt{\frac{1}{2} \mathcal{D}_{\text{KL}}(\nu_0 \| \mu_{J, h})} e^{-C_0 t}.$$

With the product initialization, we have by Lemma 4.10(2) (applied to the whole matrix) that  $\mathcal{D}_{\text{KL}}(\nu_0 \| \mu_{J, h}) \leq \|J\| n \leq 2n$ . Hence there exists a constant  $C'_2$  such that if  $C_2 = C'_2 A/c$ , then with  $T = C_2 \ln(\frac{n}{\varepsilon}) \frac{n}{s}$  steps,  $\mathcal{D}_{\text{TV}}(\nu_T \| \mu_{J, h}) \leq \frac{\varepsilon}{n^A}$ . By Lemma 4.10(1), for the Ising model  $\mu_{J_{R \times R}, h}$  the conditional distribution of  $X_S$  given  $X_{R \setminus S} = y_{R \setminus S}$  is exactly the Ising model  $\mu_{J_{S \times S}, J_{S \times R \setminus S} y_{R \setminus S} + h_S}$ . Given that the conditional distributions are sampled exactly, then the only error is that from not having fully mixed, which we set to be  $\frac{\varepsilon}{n^A}$ .

This chain of coupled random variables establishes  $\mathcal{D}_{\text{TV}}(\mathcal{D}(y), \mathcal{D}(y^{(t)})) \leq \frac{\varepsilon}{n^A}$ . Moreover, for  $t > t_{\text{max}}$ ,  $\mathcal{D}_{\text{TV}}(\mathcal{D}(y^{(t)}), \mu_{J, h}) \leq \frac{\varepsilon}{2}$  by our high-probability bound, as the root node in  $y^{(t)}$  will have been replaced with a perfect sample with probability  $\geq 1 - \frac{\varepsilon}{2}$ . It remains to note that  $t_{\text{max}} = C \ln^4(\frac{n}{\varepsilon}) \max\{\|J\|_F, 1\}$  with  $C$  depending polynomially on  $A$ . Hence we can choose  $A$  such that  $\frac{t_{\text{max}} \varepsilon}{n^A} \leq \frac{\varepsilon}{2}$ , and this finishes the proof. ◀

---

## References

- 1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1198–1211, 2020. doi:10.1145/3357713.3384317.
- 2 Yeganeh Alimohammadi, Nima Anari, Kirankumar Shiragur, and Thuy-Duong Vuong. Fractionally log-concave and sector-stable polynomials: counting planar matchings and more. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 433–446, 2021. doi:10.1145/3406325.3451123.
- 3 Jason M Altschuler and Sinho Chewi. Faster high-accuracy log-concave sampling via algorithmic warm starts. *arXiv preprint*, 2023. arXiv:2302.10249.


- 4 Nima Anari, Callum Burgess, Kevin Tian, and Thuy-Duong Vuong. Quadratic speedups in parallel sampling from determinantal distributions. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 367–377, 2023. doi:10.1145/3558481.3591104.
- 5 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy, and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE, 2018. doi:10.1109/FOCS.2018.00013.
- 6 Nima Anari, Nathan Hu, Amin Saberi, and Aaron Schild. Sampling arborescences in parallel. *arXiv preprint*, 2020. arXiv:2012.09502.
- 7 Nima Anari, Yizhi Huang, Tianyu Liu, Thuy-Duong Vuong, Brian Xu, and Katherine Yu. Parallel discrete sampling via continuous walks. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 103–116, 2023. doi:10.1145/3564246.3585207.
- 8 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence i: Modified log-sobolev inequalities for fractionally log-concave distributions and high-temperature ising models. *arXiv preprint*, 2021. arXiv:2106.04105.
- 9 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Universality of spectral independence with applications to fast mixing in spin glasses. *arXiv preprint*, 2023. arXiv:2307.10466.
- 10 Roland Bauerschmidt and Thierry Bodineau. A very simple proof of the lsi for high temperature spin systems. *Journal of Functional Analysis*, 276(8):2582–2588, 2019.
- 11 Dina Bitton, David J DeWitt, David K Hsaio, and Jaishankar Menon. A taxonomy of parallel sorting. *ACM Computing Surveys (CSUR)*, 16(3):287–318, 1984. doi:10.1145/2514.2516.
- 12 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3670–3692. SIAM, 2022. doi:10.1137/1.9781611977073.145.
- 13 Alexandre Bristiel and Pietro Caputo. Entropy inequalities for random walks and permutations. *arXiv preprint*, 2021. arXiv:2109.06009.
- 14 Djalil Chafaï. Entropies, convexity, and functional inequalities, on  $\phi$ -entropies and  $\phi$ -sobolev inequalities. *Journal of Mathematics of Kyoto University*, 44(2):325–363, 2004.
- 15 George HG Chen and R Tyrrell Rockafellar. Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997. doi:10.1137/S1052623495290179.
- 16 Yongxin Chen, Sinho Chewi, Adil Salim, and Andre Wibisono. Improved analysis for a proximal algorithm for sampling. In *Conference on Learning Theory*, pages 2984–3014. PMLR, 2022. URL: <https://proceedings.mlr.press/v178/chen22c.html>.
- 17 Yuansi Chen. An almost constant lower bound of the isoperimetric coefficient in the kls conjecture. *Geometric and Functional Analysis*, 31:34–61, 2021.
- 18 Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for markov chains. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–122. IEEE, 2022.
- 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*, pages 1537–1550, 2021. doi:10.1145/3406325.3451035.
- 20 Mary Cryan, Heng Guo, and Giorgos Mousa. Modified log-sobolev inequalities for strongly log-concave distributions. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1358–1370. IEEE, 2019. doi:10.1109/FOCS.2019.00083.
- 21 Persi Diaconis and Mehrdad Shahshahani. Time to reach stationarity in the bernoulli–laplace diffusion model. *SIAM Journal on Mathematical Analysis*, 18(1):208–218, 1987.
- 22 Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Sampling from the sherrington-kirkpatrick gibbs measure via algorithmic stochastic localization. In *2022 IEEE 63rd Annual*

- Symposium on Foundations of Computer Science (FOCS)*, pages 323–334. IEEE, 2022. doi:10.1109/FOCS54457.2022.00038.
- 23 Ronen Eldan. Thin shell implies spectral gap up to polylog via a stochastic localization scheme. *Geometric and Functional Analysis*, 23(2):532–569, 2013.
  - 24 Ronen Eldan, Frederic Koehler, and Ofer Zeitouni. A spectral condition for spectral gap: fast mixing in high-temperature ising models. *Probability theory and related fields*, 182(3-4):1035–1051, 2022.
  - 25 Jiaojiao Fan, Bo Yuan, and Yongxin Chen. Improved dimension dependence of a proximal algorithm for sampling. *arXiv preprint*, 2023. arXiv:2302.10081.
  - 26 Weiming Feng, Thomas P Hayes, and Yitong Yin. Distributed metropolis sampler with optimal parallelism. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2121–2140. SIAM, 2021. doi:10.1137/1.9781611976465.127.
  - 27 Yuval Filmus, Ryan O’Donnell, and Xinyu Wu. Log-sobolev inequality for the multislice, with applications. *Electronic Journal of Probability*, 27:1–30, 2022.
  - 28 David A Freedman. On tail probabilities for martingales. *the Annals of Probability*, pages 100–118, 1975.
  - 29 Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pages 3831–3841. PMLR, 2021. URL: <http://proceedings.mlr.press/v139/grathwohl21a.html>.
  - 30 John Hubbard. Calculation of partition functions. *Physical Review Letters*, 3(2):77, 1959.
  - 31 Vishesh Jain, Frederic Koehler, and Andrej Risteski. Mean-field approximation, convex hierarchies, and the optimality of correlation rounding: a unified perspective. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1226–1236, 2019. doi:10.1145/3313276.3316299.
  - 32 Frederic Koehler, Holden Lee, and Andrej Risteski. Sampling approximately low-rank ising models: Mcmc meets variational methods. In *Conference on Learning Theory*, pages 4945–4988. PMLR, 2022. URL: <https://proceedings.mlr.press/v178/koehler22a.html>.
  - 33 Tzong-Yow Lee and Horng-Tzer Yau. Logarithmic sobolev inequality for some models of random walks. *The Annals of Probability*, 26(4):1855–1873, 1998.
  - 34 Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Structured logconcave sampling with a restricted gaussian oracle. In *Conference on Learning Theory*, pages 2993–3050. PMLR, 2021. URL: <http://proceedings.mlr.press/v134/lee21a.html>.
  - 35 Hongyang Liu and Yitong Yin. Simple parallel algorithms for single-site dynamics. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1431–1444, 2022. doi:10.1145/3519935.3519999.
  - 36 Nicolas Loizou, Hugo Berard, Gauthier Gidel, Ioannis Mitliagkas, and Simon Lacoste-Julien. Stochastic gradient descent-ascent and consensus optimization for smooth games: Convergence analysis under expected co-coercivity. *Advances in Neural Information Processing Systems*, 34:19095–19108, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/9f96f36b7aae3b1ff847c26ac94c604e-Abstract.html>.
  - 37 Neal Madras and Dana Randall. Markov chain decomposition for convergence rate analysis. *Annals of Applied Probability*, pages 581–606, 2002.
  - 38 Andrea Montanari. Sampling, diffusions, and stochastic localization. *arXiv preprint*, 2023. arXiv:2305.10690.
  - 39 Andrea Montanari and Yuchen Wu. Posterior sampling from the spiked models via diffusion processes. *arXiv preprint*, 2023. arXiv:2304.11449.
  - 40 Ravi Montenegro and Prasad Tetali. Mathematical aspects of mixing times in markov chains. *Foundations and Trends® in Theoretical Computer Science*, 1(3):237–354, 2006. doi:10.1561/0400000003.


- 41 Izhar Oppenheim. Local spectral expansion approach to high dimensional expanders part I: Descent of spectral gaps. *Discrete & Computational Geometry*, 59(2):293–330, 2018. doi: 10.1007/s00454-017-9948-x.
- 42 Benjamin Rhodes and Michael Gutmann. Enhanced gradient-based mcmc in discrete spaces. *arXiv preprint*, 2022. arXiv:2208.00040.
- 43 Justin Salez. A sharp log-Sobolev inequality for the multislice. *Ann. H. Lebesgue*, 4:1143–1161, 2021. doi:10.5802/ahl.99.
- 44 Holger Sambale and Arthur Sinulis. Modified log-sobolev inequalities and two-level concentration. *arXiv preprint*, 2019. arXiv:1905.06137.
- 45 Ruoqi Shen and Yin Tat Lee. The randomized midpoint method for log-concave sampling. *Advances in Neural Information Processing Systems*, 32, 2019.
- 46 Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. URL: <http://proceedings.mlr.press/v37/sohl-dickstein15.html>.
- 47 Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.
- 48 Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint*, 2020. arXiv:2011.13456.
- 49 Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.
- 50 Ruqi Zhang, Xingchao Liu, and Qiang Liu. A langevin-like sampler for discrete distributions. In *International Conference on Machine Learning*, pages 26375–26396. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/zhang22t.html>.



# Towards Simpler Sorting Networks and Monotone Circuits for Majority

Natalia Dobrokhotova-Maikova ✉ 

Yandex, Tel Aviv, Israel

Alexander Kozachinskiy ✉ 

IMFD & CENIA, Santiago, Chile

Vladimir Podolskii ✉ 

Tufts University, Medford, MA, USA

---

## Abstract

In this paper, we study the problem of computing the majority function by low-depth monotone circuits and a related problem of constructing low-depth sorting networks. We consider both the classical setting with elementary operations of arity 2 and the generalized setting with operations of arity  $k$ , where  $k$  is a parameter. For both problems and both settings, there are various constructions known, the minimal known depth being logarithmic. However, there is currently no known efficient deterministic construction that simultaneously achieves sub-log-squared depth, simplicity, and has a potential to be used in practice. In this paper we make progress towards resolution of this problem.

For computing majority by standard monotone circuits (gates of arity 2) we provide an explicit monotone circuit of depth  $O(\log_2^{5/3} n)$ . The construction is a combination of several known and not too complicated ideas. Essentially, for this result we gradually derandomize the construction of Valiant (1984).

As one of the intermediate steps in our result we need an efficient construction of a sorting network with gates of arity  $k$  for arbitrary fixed  $k$ . For this we provide a new sorting network architecture inspired by representation of inputs as a high-dimensional cube. As a result we obtain a simple construction that improves previous upper bound of  $4\log_k^2 n$  to  $2\log_k^2 n$ . We prove the similar bound for the depth of the circuit computing majority of  $n$  bits consisting of gates computing majority of  $k$  bits. Note, that for both problems there is an explicit construction of depth  $O(\log_k n)$  known, but the construction is complicated and the constant hidden in  $O$ -notation is huge.

**2012 ACM Subject Classification** Theory of computation → Models of computation

**Keywords and phrases** Sorting networks, constant depth, lower bounds, threshold circuits

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.50

**Category** RANDOM

**Funding** *Alexander Kozachinskiy*: funded by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID, and by the Millennium Science Initiative Program – Code ICN17002.

## 1 Introduction

More than 50 years ago, Foster and Stockton [10] devised, in modern terms, an  $O(\log n)$ -depth Boolean circuits (with AND and OR gates of fan-in 2 and also NOT gates) that, given  $n$  input bits, computes the binary representation of their sum. Their construction is explicit, that is, the circuit can be computed in deterministic  $n^{O(1)}$ -time. Moreover, it is relatively simple, where the main trick is to compute in *constant* depth, for any 3 numbers, given in binary, 2 numbers with the same sum, also given in binary. In about  $\log_{3/2} n$  steps, we get from  $n$  input bits to just two numbers, both  $O(\log n)$ -bit long. After that, one can compute their sum in depth  $O(\log n)$ , say, by the grade school algorithm.



© Natalia Dobrokhotova-Maikova, Alexander Kozachinskiy, and Vladimir Podolskii; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 50; pp. 50:1–50:18



Leibniz International Proceedings in Informatics

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

An easy consequence of this [20] is that all *symmetric* Boolean functions (those whose value is determined by the number of 1s in the input) are in the class  $NC^1$ , that is, can be computed by  $O(\log n)$ -depth fan-in 2 Boolean circuits. It should be noted, however, that in this construction, it is unavoidable to use NOT gates, even for symmetric functions that are monotone, like the *majority function*, outputting 1 if and only if more than half of the input bits are 1's. This is because inside the construction we compute the sum of input bits in binary, and the digits of this sum are non-monotone functions. Thus, if we want, say, a *monotone* Boolean circuit (only fan-in 2 AND and OR gates with no NOT gates) for the majority function that has depth  $O(\log n)$ , we need other ideas.

Monotone Boolean circuits of depth  $O(\log n)$  for the majority function are known to exist, but to this day, there is no explicit construction as simple as the construction of Foster and Stockton. Namely, there are two constructions, one is extremely simple but randomized (due to Valiant [32]), and the other is explicit but extremely involved (due to Ajtai, Komlós and Szemerédy [1]).

One can represent the construction of Valiant as a ternary tree of depth  $C \cdot \log n$ , where every node computes the majority of its three children [12]. As for the leaves, we simply put a random input variable to every leaf in the tree, independently for different leaves. It is relatively easy to show that, for every input  $x \in \{0, 1\}^n$  and for a large enough absolute constant  $C > 0$ , the probability that the tree computes the value of the majority function on  $x$  is larger than  $1 - 2^{-n}$ . Thus, there exists a choice of putting input variables to leaves that gives us a circuit, computing the majority function for all inputs.

*Derandomizing* the Valiant's construction seems a tempting approach for constructing an explicit  $O(\log n)$ -depth monotone circuit for the majority function. Nevertheless, there has been limited progress in this direction. Hoory, Magen and Pitassi [13] have improved the size of the Valiant's construction, but their construction is still randomized. In turn, Cohen et al. [7] observed how to use hash functions to reduce the number of random bits to  $O(\log n)$  but at the cost of having the probability of error about  $1/\text{polylog}(n)$ , which is not enough for the complete derandomization.

Using a completely different approach, Ajtai, Komlós and Szemerédy [1] constructed an explicit (computable in deterministic polynomial time)  $O(\log n)$ -depth monotone circuit for the majority function. It is worth mentioning that the approach of [1] uses randomness as well, however, they derandomize their construction on the later steps. In fact, they did much more than that – they constructed an explicit *sorting network* with  $O(\log n)$  layers. A sorting network receives on input an array with  $n$  numbers and outputs the same array but with numbers going in the non-decreasing order. In each layer, there is a fixed partition of the entries of the array into pairs, where to each pair one applies a *comparator*, swapping the numbers in a pair if they are not in the non-decreasing order. A sorting network can be easily turned into a monotone circuit for the majority function whose depth is equal to the number of layers of the network. This is because on binary inputs, the comparator can be simulated with one AND gate and one OR gate. The value of the majority function then can be found as the value of the median entry of the sorted array.

For sorting networks, there are several simple and practical constructions with  $\Theta(\log^2 n)$  layers [16, 2, 22]. The construction of Ajtai, Komlós and Szemerédy [1], usually referred to as the AKS sorting network, has  $O(\log n)$  layers, which is asymptotically optimal. However, despite some further simplifications by Paterson [24] and Seiferas [26], this construction is famously very involved with a constant above 1000 before  $\log n$ . As for the lower bounds, there is a folklore  $(2 - o(1)) \log_2 n$  lower bound on the number of layers for networks sorting  $n$  numbers. It was improved by Yao [34] and later by Kahale et al. [15] with the current record about  $3.27 \log_2 n$ .

For constructing just an explicit  $O(\log n)$ -depth monotone circuit for the majority function, we do not necessarily have to construct a sorting network. This gives us a hope that the difficulty of the AKS construction can be avoided. In this paper, we make a progress in this direction, giving the first explicit monotone circuit for the majority function that has sub- $O(\log^2 n)$  depth while not using the AKS methodology at all.

► **Theorem 1.** *There is a polynomial time constructable monotone circuit for  $\text{MAJ}_n$  of polynomial size and depth  $O(\log^{5/3} n)$ .*

Our proof combines several relatively simple steps. The principle component is a *partial derandomization* of the Valiant's construction, using some ideas of Cohen et al. [7] but with different setting of parameters. Next, we repeatedly apply two operations to the resulting randomized circuit. The first operation is a brute-force derandomization, that searches through all possible random bits of the randomized circuit. The second one is a composition with a circuit for  $\text{MAJ}_n$  that consists of  $\text{MAJ}_k$  gates and has depth  $O(\log_k^2 n)$ . Existence of such circuits is known [23, 8], but in this paper, we also give a new simple construction, based on alternative ideas and with better constant before  $\log_k^2 n$ .

In fact, all known construction of such circuits, including a new one from this paper, come from sorting networks with comparators that can simultaneously sort  $k > 2$  positions of the array. We will call them  $k$ -sorting networks. They appear in the literature since the 70s, the setting is mentioned already in the Knuth's book [16, Problem 5.3.4.54], followed by numerous works [30, 23, 3, 21, 8, 18, 28, 11, 35]. They are usually studied to better understand the structure of ordinary sorting networks (for example, a version of AKS sorting network with improved constant relies on  $k$ -sorting network in intermediate constructions [6]). In particular,  $k$ -sorting networks are closely related to recursive constructions of sorting networks. Having a good construction of a  $k$ -sorting network, one can apply it to its own comparators, getting a construction with smaller  $k$ , until eventually  $k$  becomes 2, and we get an ordinary sorting network.

Chvátal shows in his lecture notes [6], the AKS sorting network also generalizes to this setting, giving a construction of depth  $O(\log_k n)$ . However, as with the AKS sorting network itself, this construction is complicated and impractical. In fact, even constructing a  $k$ -network of depth  $O(\log_k^2 n)$  is significantly harder for general  $k$  than just for  $k = 2$ . Standard  $O(\log_2^2 n)$  constructions for 2-sorting networks are based on divide and conquer approach, in which we first recursively sort parts of input and then merge sorted parts together. For the case of  $k$ -sorting networks to get a network of depth  $O(\log_k^2 n)$  with the same approach merging step needs to merge many parts simultaneously and this task becomes non-trivial [19, 27, 25, 5]. We are aware of just two constructions like that: Cypher and Sanz [8] gave a simple and potentially practical  $k$ -sorting network of depth  $\leq 5 \log_k^2 n$  (for  $k \geq \log^4 n$ ) and Parker and Parbery [23] gave a construction of depth  $\leq 4 \log_k^2 n$  (in case when  $n$  is an integral power of  $k$ ). As for the lower bounds, any  $k$ -sorting network with  $n$  inputs must have depth at least  $\log_k n$  because otherwise outputs cannot be connected to all  $n$  inputs. Dobrokhotova-Maikova et al. [9] improved this bound to roughly  $2 \log_k n$ . They also found optimal values of  $k$  for small values of depth  $d$ . More specifically, for sorting networks of depth  $d = 1, 2$  they show that  $k$  cannot be smaller than  $n$ , for  $d = 3$  the optimal value is  $k = \lceil \frac{n}{2} \rceil$  and for  $d = 4$  the optimal value is  $k = \Theta(n^{2/3})$ . These results indicate that small depth  $k$ -sorting networks are not enough for iterative approach to sub-log-squared sorting network and we need either good  $k$ -sorting network constructions of depth greater than 4 or additional ideas.

Our second result is a new architecture for  $k$ -sorting networks. An application of this architecture is a  $k$ -sorting network of depth  $2 \log_k^2 n$ , improving the constant compared to the results of [8, 23]. More precisely, we prove the following theorem.

► **Theorem 2.** *For any  $n$  and for any  $k$  such that  $\log k = \omega(\log \log n)$  (or, to put it differently,  $k$  is growing faster than any  $\text{polylog}(n)$ ), there exists a  $k$ -sorting network of depth at most  $(2 + o(1)) \log_k^2 n$ . The sorting network can be computed in polynomial time.*

The key idea behind this construction is to represent the input array as a hypercube of high dimension and sort various sections of this cube. We note that the idea of representing an array as a multidimensional structure is not new, for example, Leighton [19] in his ColumnSort represented the array as a two dimensional table and Cypher and Sanz [8] use a representation of larger dimension. In our construction it is important that we use the dimension greater than 2 and that the sections of the cube that are used for sorting have non-trivial intersection. On the conceptual level, the main novelty in our construction is the notion of  $s$ -sorting. We call the array  $s$ -sorted if the whole array is sorted correctly apart from some interval of length at most  $s$ . Most (if not all) log-squared-depth sorting network constructions adopt the divide and conquer strategy. The  $O(\log_k^2 n)$ -depth construction in [23] is not an exception, to sort an array of size  $n$ , they split it into subarrays of size  $n/k$ , sort them recursively and merge them afterward. However, merging  $k$  subarrays using  $k$ -sorting network is relatively expensive. To improve over previous construction, we work with  $s$ -sorted subarrays instead. We show how to merge them effectively (using the hypercube idea) and then show how we can build a recursive construction based on them.

It is not hard to see that all outputs of a comparator with  $k$  inputs can be computed by  $\text{MAJ}_{2k}$  gates. This means that Theorem 2 yields an  $(2 + o(1)) \log_k^2 n$ -depth circuit for  $\text{MAJ}_n$ , consisting of  $\text{MAJ}_k$  gates, which is a final component for our construction in Theorem 1

To additionally illustrate applications of our construction, we consider constant depth sorting networks and circuits for majority. We show that there is a depth-4  $\text{MAJ}_k$ -circuit for  $\text{MAJ}_n$  for  $k = O(n^{3/5})$ . As another application we address the question of  $k$ -sorting networks for  $k = O(n^{1/2})$ . In [16] Knuth posed a problem of constructing a minimal depth  $k$ -sorting network for the input of size  $k^2$ . Parker and Parbery [23] gave a construction of depth 9. We improve this to depth 8 at the cost of using comparators of size  $O(k)$  for  $k^2$  input size. The results of [9] show that the depth of such a network must be at least 5.

The rest of the paper is organized as follows. In Section 2 we provide necessary preliminary information. In Section 3 we construct a monotone circuit for majority of depth  $O(\log^{5/3} n)$ . In Section 4 we provide a new construction of  $k$ -sorting networks and deduce the corollaries. In Section 5 we discuss some open problems.

## 2 Preliminaries

We use the standard notation  $[n] = \{1, \dots, n\}$ . We sometimes omit the base of the logarithms, by default we assume that the base is 2.

### 2.1 Sorting Networks

A depth- $d$   $k$ -sorting network with  $n$  inputs consists of  $d + 1$  arrays  $A_1, \dots, A_{d+1}$ , each of length  $n$ . Between any two arrays  $A_i$  and  $A_{i+1}$  there is a *layer of comparators* (the first layer is between  $A_1$  and  $A_2$ , the second layer is between  $A_2$  and  $A_3$ , and so on). A layer of comparators is a partition of the set  $\{1, 2, \dots, n\}$  into subsets of size at most  $k$  called *comparators*.

The input is given in an array  $A_1$  and all other arrays are computed by the network one by one in the following way. If  $S \subseteq [n]$  is a comparator from the  $i$ th layer, then it is applied to the entries  $\{A_i[j] \mid j \in S\}$ . It sorts their values in the non-decreasing order and puts the results into the entries  $\{A_{i+1}[j] \in A_{i+1} \mid j \in S\}$ . We say that a network is *sorting* if for any input  $A_1$  the array  $A_{d+1}$  is sorted.

We reserve the name *sorting network* for 2-sorting networks.

It is well known that to check that the sorting network sorts all possible inputs, it is enough to check that it sorts just 0/1-inputs.

► **Lemma 3** (Zero-one principle [16]). *A network with  $n$  inputs sorts all integer sequences in the non-decreasing order if and only if it sorts all sequences from  $\{0, 1\}^n$  in the non-decreasing order.*

By this principle, when constructing sorting networks, we can assume that each input cell receives either 0 or 1.

The following simple observation will be useful to us.

► **Lemma 4.** *If the  $t$  largest or the  $t$  smallest entries in the array are positioned correctly (i.e., in the last  $t$  cells and in the first  $t$  cells, respectively), then after the application of several comparators they are still positioned correctly.*

**Proof.** We can show by induction on  $i$  that the smallest and the largest entries do not move if they are already positioned correctly. The key observation is that if some of these entries are inputted into one of the comparators  $S$ , they will not be moved. ◀

## 2.2 From Sorting Networks to Majority Circuits

We use the standard notion of Boolean circuits (see, e.g. [14]). As inputs, we allow Boolean variables and Boolean constants 0 and 1. The size of the circuit is the number of gates in it.

Given a  $k$ -sorting network we can get a circuit computing majority from it. More specifically, restrict the inputs to the network to  $\{0, 1\}^n$  and consider one  $k$ -comparator  $S$ . Note that its  $k$ th output is equal to 1 if and only if there is at least one 1 in the input. In other words, the  $k$ th output is equal to OR of input bits. Its  $(k - 1)$ th output is equal to 1 if and only if there are at least two 1s in the input. More generally, it is easy to see that the  $(k - i)$ th output of the  $k$ -comparator outputs a threshold function

$$\text{THR}_k^i(x) = \begin{cases} 1 & \text{if } |x| > i, \\ 0 & \text{otherwise,} \end{cases}$$

where  $|x|$  denotes the weight of the vector  $x \in \{0, 1\}^k$ , that is, the number of 1s in it. We reserve the notation  $\text{MAJ}_k(x)$  for the function  $\text{THR}_k^{k/2}(x)$ .

We can substitute each comparator in the network by  $k$  majority functions. Note that by adding several constants 0 or 1 as inputs to the gate we can convert any  $\text{THR}_k^i$  function into  $\text{MAJ}_{k'}$  with  $k' \leq 2k$ .

Now, it remains to observe that the median bit in the output array computes exactly  $\text{MAJ}_n$ . Thus, as a result, we get the following lemma.

► **Lemma 5.** *Any  $k$ -sorting network of depth  $d$  and size  $s$  can be effectively converted into a circuit of depth  $d$  and size  $ks$  consisting of  $\text{MAJ}_{2k}$  gates and computing majority. In the case  $k = 2$ , we get just a monotone circuit consisting of AND and OR.*

## 2.3 Approximate Majority

By  $\varepsilon$ -approximate majority function  $\text{MAJ}_n^\varepsilon$  we denote the partial function that outputs  $\text{MAJ}_n$  of its input but is defined only on the inputs where the fraction of ones in it is bounded away by  $\varepsilon$  from  $1/2$ .

We need the following result by Viola [33] which can be viewed as a derandomization of the Sipser–Lautemann theorem [29, 17]

► **Theorem 6** ([33]). For any constant  $\varepsilon > 0$ , one can compute  $\text{MAJ}_n^\varepsilon$  explicitly by a monotone circuit of size  $\text{poly}(n)$  and depth  $O(\log n)$ .

(Monotonocity condition is implicit in [33] but easily observable from the construction).

### 2.4 $t$ -Wise Independent Hash Functions

We need the notion of  $t$ -wise independent hash functions.

► **Definition 7.** For integers  $N$  and  $t$  such that  $t \leq N$ , a family of function  $\mathcal{H} = \{h: [N] \rightarrow [N]\}$  is  $t$ -wise independent if for all distinct  $x_1, \dots, x_t \in [N]$  the random variables  $h(x_1), \dots, h(x_t)$  are independent and uniformly distributed in  $[N]$ , when  $h \in \mathcal{H}$  is drawn uniformly.

► **Theorem 8** ([31]). For every integer  $n$  and  $t$  such that  $t \leq 2^n$  there is a family of  $t$ -wise independent functions  $\mathcal{H} = \{h: \{0, 1\}^n \rightarrow \{0, 1\}^n\}$  such that choosing a random function from  $\mathcal{H}$  takes  $nt$  random bits and evaluating a function from  $\mathcal{H}$  takes time  $\text{poly}(n, t)$ .

► **Theorem 9** ([4]). Let  $X$  be the average of  $N$   $t$ -wise independent random variables  $X_1, \dots, X_N \in [0, 1]$  for even  $t$ . Then for any  $\varepsilon > 0$  we have

$$\Pr [|X - \mathbb{E}[X]| \geq \varepsilon] \leq 1.1 \left( \frac{t}{N\varepsilon^2} \right)^{t/2}.$$

## 3 Sub-log-squared Circuit for Majority

In this section, we provide a proof of Theorem 1.

Our goal is to compute  $\text{MAJ}_n$  by an explicit circuit of polynomial size and  $o(\log^2 n)$  depth. We assume for convenience that  $n$  is odd (for even  $n$  we can consider a circuit for  $n + 1$  and substitute one variable by a constant). We start with some inferior circuit and perform several operations that allow us to gradually improve the parameters. However, on our way, we need to consider randomized circuits as well, and apart from size and depth, we will also be interested in the number of random bits and the error probability. More specifically, a circuit is an  $(s, d, r, \text{err})$ -circuit for majority if its size is at most  $2^s$ , depth is at most  $d$ , we can construct a circuit using at most  $r$  random bits and the error probability on each input is at most  $2^{-\text{err}}$ . Here all parameters are functions in the number of inputs  $n$  (we write  $\text{err} = \infty$  when the circuit is correct with probability 1). All circuits we are going to consider are effectively constructible: there is an algorithm that given the values of random bits constructs a circuit in polynomial time in the size of the circuit.

Given a circuit with some parameters, we will use two operations to obtain new circuits. We are introducing these operations in the next two lemmas. Their effect on the circuit is summarized in the table below.

| Initial circuit | Brute-force derandomization | Downward self-reduction  |
|-----------------|-----------------------------|--|
| $s(n)$          | $O(s(n) + r(n))$            | $O(\log n) + s(2k)$  |
| $d(n)$          | $d(n) + O(r(n))$            | $O\left(\left(\frac{\log_2 n}{\log_2 k}\right)^2 d(2k)\right)$ |
| $r(n)$          | 0                           | $r(2k)$  |
| $\text{err}(n)$ | $\infty$                    | $\text{err}(2k) - O(\log n)$                                   |

► **Lemma 10** (Brute-force derandomization). *If there is an  $(s, d, r, 2)$ -circuit  $C$ , then there is an  $(O(s + r), d + O(r), 0, \infty)$ -circuit.*

This lemma allows us to get rid of randomness but increases the depth and the size of the circuit if  $r$  is large.

**Proof.** Consider a randomized circuit  $C_y(x)$ , where  $x \in \{0, 1\}^n$  is an input and  $y \in \{0, 1\}^r$  is the sequence of random bits. Assume  $C_y(x)$  has the parameters, as in the statement of the lemma. Consider circuits  $C_y(x)$  for all possible values of  $y$  and observe that for any  $x$  the fraction of circuits that output  $\text{MAJ}_n(x)$  is at least  $1 - 1/4 = 3/4$ . Thus, if we feed  $C_y(x)$  for all  $y$  into a circuit from Theorem 6 computing  $\text{MAJ}_{2^r}^\varepsilon$ , the output is exactly  $\text{MAJ}_n(x)$ .

The size of the resulting circuit is at most  $2^r \cdot 2^s + \text{poly}(2^r)$ , where the first term corresponds to computing  $C_y(x)$  for all  $y$  and the second term corresponds to computing  $\text{MAJ}_{2^r}^\varepsilon$ . Thus, the size is  $2^{O(s+r)}$ . Since all  $C_y(x)$  can be computed in parallel, the depth of the circuit is at most  $d + O(r)$ . The resulting circuit does not use random bits and is always correct. ◀

► **Lemma 11** (Downward self-reduction). *If there is an  $(s(n), d(n), r(n), \text{err}(n))$ -circuit  $C$ , then for any  $k < n$  there is an  $(O(\log n) + s(2k), O(\log_k^2 n \cdot d(2k)), r(2k), \text{err}(2k) - O(\log n))$ -circuit.*

This operation increases the depth (if  $d(n)$  is sub-log-squared), but allows to reduce other parameters.

**Proof.** Consider a  $k$ -sorting network of depth  $O(\log_k^2 n)$ , given by [23] or by our Theorem 2 (the latter allows only for limited values of  $k$ , but the values we will actually use in the construction below are within the limits). By Lemma 5 this network gives us a monotone circuit with the same parameters consisting of  $\text{MAJ}_{2k}$  gates computing  $\text{MAJ}_n$ , denote this circuit by  $C(x)$ , where  $x \in \{0, 1\}^n$ .

Consider a  $(s(2k), d(2k), r(2k), \text{err}(2k))$ -circuit  $C_y$  on  $k$  inputs, where  $y \in \{0, 1\}^{r(2k)}$ . Fix  $y$  and substitute each  $\text{MAJ}_{2k}$  gate in  $C$  by  $C_y$ . Denote the resulting circuit by  $D_y(x)$ . This is a standard monotone Boolean circuit, its size is  $\text{poly}(n) \cdot 2^{s(2k)}$ , its depth is  $O(\log_k^2 n \cdot d(2k))$  and the number of random bits is  $r(2k)$ .

It remains to show that the error probability is not too large. For this fix some input  $x \in \{0, 1\}^n$ . Consider all  $\text{MAJ}_{2k}$  gates in  $C(x)$  and denote their inputs when  $x$  is fed to  $C$  by  $z^1, z^2, \dots, z^t$ . Here  $t$  is the size of  $C$  and is polynomial in  $n$ .

For each  $z^i$  the probability over random  $y$  that  $C_y(z^i)$  computes  $\text{MAJ}_{2k}(z^i)$  incorrectly is at most  $2^{-\text{err}(2k)}$ . By union bound, with probability at least  $1 - t2^{-\text{err}(2k)}$  we have  $C_y(z^i) = \text{MAJ}_k(z^i)$  for all  $i$  and thus  $D_y(x)$  computes  $\text{MAJ}_n(x)$  correctly. Thus, the probability of error of the resulting circuit is at most

$$t \cdot 2^{-\text{err}(2k)} = 2^{-\text{err}(2k) + O(\log n)}.$$

Now we describe our starting circuit. Interestingly, it is constructed as a partial derandomization of Valiant's construction.

► **Lemma 12.** *There is an explicit circuit for majority with parameters  $(O(\log n), O(\log n), O(\log^3 n), \Omega(\log^2 n))$ .*

We provide the proof of Lemma 12 in Section 3.1 below, but before that, we explain how to finish the construction of the desired circuit for  $\text{MAJ}_n$ .

Starting with the circuit provided by Lemma 12, we first apply downward self-reduction with the parameter  $k$  satisfying  $\log k = C\sqrt{\log n}$  for some big enough constant  $C > 0$ , then we apply brute-force derandomization, and then we apply downward self-reduction again with  $k$  satisfying  $\log k = \log^{2/3} n$ . We summarize the changes in the parameters after each step in the table below.

|                 | Initial circuit    | Step 1   | Step 2                         | Step 3  |
|-----------------|--------------------|--|--------------------------------|---|
|                 |                    | Self-reduction<br>with<br>$\log k = \sqrt{\log n}$ | Brute-force<br>derandomization | Self-reduction<br>with<br>$\log k = \log^{2/3} n$ |
| $s(n)$          | $O(\log n)$        | $O(\log n)$  | $O(\log^{3/2} n)$              | $O(\log n)$                                       |
| $d(n)$          | $O(\log n)$        | $O(\log^{3/2} n)$                                  | $O(\log^{3/2} n)$              | $O(\log^{5/3} n)$                                 |
| $r(n)$          | $O(\log^3 n)$      | $O(\log^{3/2} n)$                                  | 0                              | 0   |
| $\text{err}(n)$ | $\Omega(\log^2 n)$ | $\Omega(\log n)$                                   | $\infty$                       | $\infty$  |

► Remark 13. Note that with the two operations in hand, there are not that many options to apply them to a given initial construction. It is not hard to check that applying downward self-reduction two times in a row is not better than applying it once with the appropriate value of  $k$ . Clearly, there is no need to apply the derandomization step twice. From this, it is not hard to see that our sequence of operations is actually optimal. Once the optimal sequence of operations is established, it is not hard to check that our choice of parameters in downward self-reductions is optimal as well.

### 3.1 Proof of Lemma 12

In this subsection, we are going to prove Lemma 12. The high-level idea is to partially derandomize Valiant's construction. To make the presentation self-contained we first recall the idea behind this construction.

Suppose we have independent random bits  $x, y, z$  that are equal to 1 with probability  $p$  and consider  $\text{MAJ}_3(x, y, z)$ . It is not hard to see that it outputs 1 with probability  $f(p) = p^3 + 3p^2(1-p)$ . Consider  $p = \frac{1}{2} + \varepsilon$  for some  $\varepsilon > 0$  and denote  $\varepsilon' = f(p) - \frac{1}{2}$ . Then

$$\varepsilon' = f(p) - \frac{1}{2} = f(p) - f\left(\frac{1}{2}\right) = f'(\alpha) \left(p - \frac{1}{2}\right) = f'(\alpha)\varepsilon$$

for some  $\alpha \in [\frac{1}{2}, p]$ . Note that  $f'(p) = 6p - 6p^2 = 6p(1-p)$ . It is easy to see that for  $\alpha \in [\frac{1}{2}, \frac{2}{3}]$  we have  $f'(\alpha) \geq \frac{4}{3}$ . Thus, for  $\varepsilon \in [0, \frac{1}{6}]$  we have  $\varepsilon' \geq \frac{4}{3}\varepsilon$ .

Now, we can use this in the following way. Consider  $\text{MAJ}_n$  for odd  $n$  and consider its arbitrary input  $x$ . Without loss of generality, assume that  $\text{MAJ}_n(x) = 1$ . If we draw one variable from  $x$  uniformly at random, it is equal to 1 with probability at least  $\frac{1}{2} + \frac{1}{n}$ . Consider a  $\text{MAJ}_3$  gate and feed to it three independently and uniformly drawn input variables. By the analysis above the output of such a  $\text{MAJ}_3$  gate is equal to 1 with probability at least  $\frac{1}{2} + \frac{4}{3} \cdot \frac{1}{n}$ . Now we can repeat this: consider three such  $\text{MAJ}_3$  gates and feed their outputs to another  $\text{MAJ}_3$  gates. The result is equal to 1 with probability  $\frac{1}{2} + \left(\frac{4}{3}\right)^2 \frac{1}{n}$ . After  $O(\log n)$  many iterations, we get a  $O(\log n)$ -depth randomized circuit consisting of  $\text{MAJ}_3$  gates that output the correct value with probability at least  $\frac{2}{3}$ . Valiant's argument further improves this probability, but we will not need this part of the argument.

The randomized circuit above uses too many random bits. Now we are going to modify the construction in a way, that uses randomness more efficiently. We will use some ideas from [7].



Construct the following circuit consisting of MAJ<sub>3</sub> gates. The circuit contains  $\Theta(\log n)$  layers, each containing  $N = n^3$  gates. The bottom layer consists of input variables, each repeated  $\frac{N}{n} = n^2$  times (it is redundant to copy variables several times, we do this exclusively for the sake of uniformity of the construction). In other layers, each gate computes the MAJ<sub>3</sub> function of some gates from the previous layer. To assign the inputs to each gate, for each layer  $j$  we draw three fresh (and independent of each other)  $t$ -wise independent hash functions  $f_j, g_j, h_j: [N] \rightarrow [N]$  for  $t = \Theta(\log n)$ . For a gate with number  $i$  in layer  $j$  we set its inputs to be gates with numbers  $f_j(i), g_j(i)$  and  $h_j(i)$  in layer  $(j - 1)$ .

Before we finish the construction of the circuit, let us analyze the current part. Consider some input  $x \in \{0, 1\}^n$ , assume without loss of generality that  $\text{MAJ}_n(x) = 1$ . Denote by  $\frac{1}{2} + \varepsilon_i$  the fraction of gates on level  $i$  that output 1. For  $i = 1$  we have  $\varepsilon_1 \geq \frac{1}{n}$ .

Each gate on level  $i$  receives three independent inputs from the previous level. Thus, the probability that it outputs 1 is at least  $\frac{1}{2} + \frac{4}{3}\varepsilon_{i-1}$  (we have shown this above only for  $\varepsilon_{i-1} \leq \frac{2}{3}$ , but these values of  $\varepsilon_{i-1}$  are enough for our construction as well). Thus, the expected fraction of ones in level  $i$  is also at least  $\frac{1}{2} + \frac{4}{3}\varepsilon_{i-1}$ .

Now we would like to use concentration inequality to show that with high probability the fraction of correct values is not much smaller than its expectation. Note that once the outputs of the gates on level  $i - 1$  are fixed, the outputs of the gates on level  $i$  are  $t$ -wise independent.

Let  $\varepsilon = \frac{1}{6n}$  and denote by  $X_i$  the output of  $i$ -th gate. Then by Theorem 9 we have

$$\Pr \left[ \left| \sum_i X_i/N - \left( \frac{1}{2} + (4/3)\varepsilon_{j-1} \right) \right| \geq \varepsilon \right] \leq 1.1 \left( \frac{t}{N\varepsilon^2} \right)^{t/2} = 2^{-\Theta(\log^2 n)}.$$

By union bound the probability that on each level  $\varepsilon_j \geq \frac{4}{3}\varepsilon_{j-1} - \varepsilon$  is at least

$$1 - O(\log n) \cdot 2^{-\Theta(\log^2 n)} = 1 - 2^{-\Theta(\log^2 n)}.$$

Thus, we can show by induction on  $j$  that with probability at least  $1 - 2^{-\Theta(\log^2 n)}$  we have

$$\varepsilon_j \geq \frac{4}{3}\varepsilon_{j-1} - \varepsilon \geq \frac{7}{6}\varepsilon_{j-1} + \frac{1}{6}\varepsilon_{j-1} - \frac{1}{6n} \geq \frac{7}{6}\varepsilon_{j-1},$$

where in the last inequality we use that by induction hypothesis we have  $\varepsilon_{j-1} \geq \left(\frac{7}{6}\right)^{j-1} \cdot \varepsilon_1 \geq \frac{1}{n}$ .

Thus, just like in Valiant's argument, after  $O(\log n)$  iterations, with probability  $1 - 2^{-\Theta(\log^2 n)}$ , we have  $\varepsilon_j \geq \frac{2}{3}$ . At this point, it remains to apply to the last layer a circuit from Theorem 6.

It is easy to see that the size of the resulting circuit is  $\text{poly}(n)$ , the depth is  $O(\log n)$ , error probability is  $2^{-\Theta(\log^2 n)}$ . As for the random bits, note that in the construction we need  $O(\log n)$   $t$ -wise independent hash functions from  $[N]$  to  $[N]$ . By Theorem 8 there are families of such functions defined using  $O(t \log N)$  random bits. In total we need

$$O(\log n) \cdot O(t \log N) = O(\log^3 n)$$

random bits.

This finishes the proof of Lemma 12.

► **Remark 14.** Instead of applying a circuit for Approximate Majority to the last layer, we could do the following: sample  $m = O(\log^2 n)$  gates from the last layer uniformly at random and then compute the majority on these  $m$  gates using some simple circuit of depth  $O(\log^2 m)$ . By Chernoff's inequality, this adds at most  $2^{-\Omega(m)} = 2^{-\Theta(\log^2 n)}$  to the error probability, and we need  $O(\log^3 n)$  random bits. In turn, the increase in depth and size is negligible.

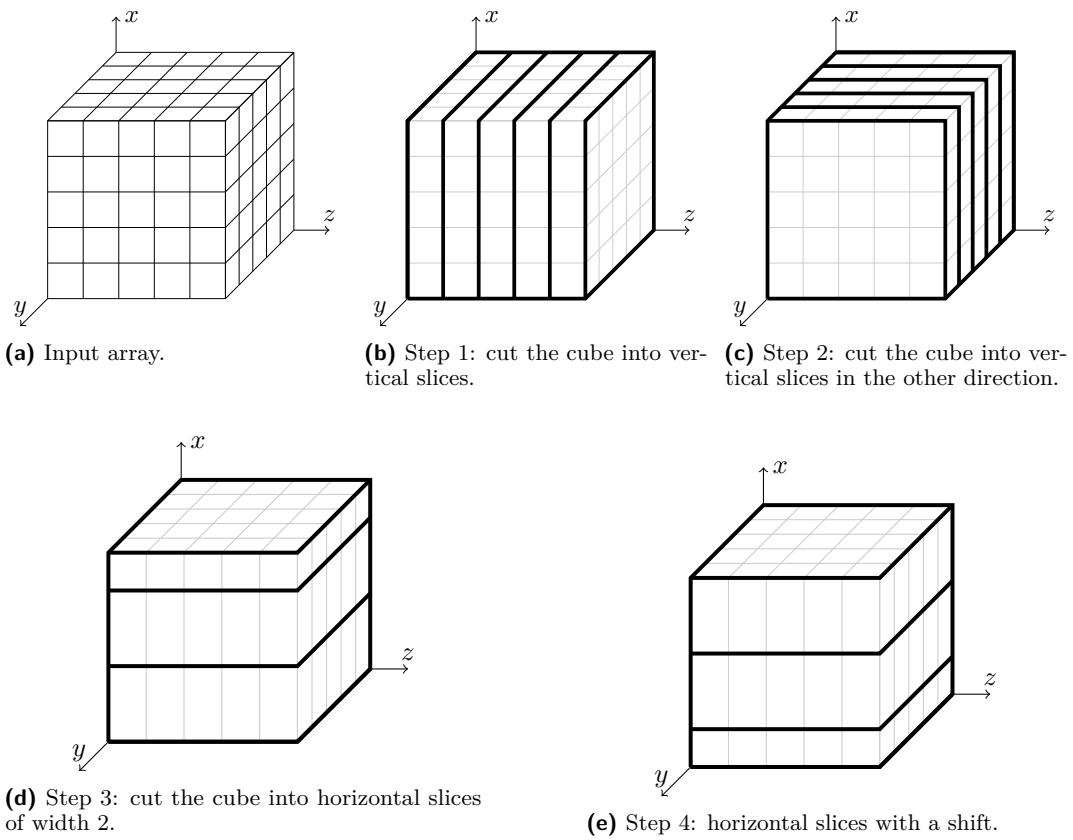
**4** *k*-Sorting Network Construction

**4.1** Proof Strategy

Before we proceed to the proof we would like to illustrate the idea considering some specific value of *k*. For convenience, we assume that *n* is a perfect cube.

► **Lemma 15.** *Assume that  $n = t^3$  for natural  $t$ . Then there is a depth-4 *k*-sorting network with  $k = 2t^2 = 2n^{2/3}$ .*

We present the proof using a geometric interpretation of an input array as a three-dimensional cube. However, note that a similar result is implicit in [19] and it is essentially the same construction, just stated in different terms. We also note that it is known that this is the optimal (up to a constant factor) value of *k* for depth-4 sorting networks [9].



■ **Figure 1** Sorting network for  $k = 2n^{2/3}$  (here  $n = 125$ ,  $k = 50$  and  $t = 5$ ).

**Proof.**

**Step 1 – construction.** We represent entries of an input array as a 3-dimensional cube with the side *t* (see Figure 1a). We place the first *t*<sup>2</sup> entries of an array in the bottom layer of the cube, the next *t*<sup>2</sup> entries in the second layer of the cube and so on. In each layer the entries are positioned row by row.

To be more precise, assume that the array *A* is enumerated as  $[a_1, \dots, a_n]$ . We reenumerate the same array as

$$[a_{111}, a_{112}, \dots, a_{11t}, a_{121}, \dots, a_{12t}, \dots, a_{ttt}].$$

That is, entries of an array are enumerated by sequences  $(x, y, z) \in \{1, \dots, t\}^3$  in the lexicographic order. In Figure 1  $a_{xyz}$  corresponds to a subcube with coordinates  $(x, y, z)$ .

In the first layer of the sorting network we split the cube into vertical slices of width 1 and feed each slice to a  $t^2$ -comparator (see Figure 1b). To be more precise, for each  $i = 1, \dots, t$  we feed entries  $a_{xyi}$  for all  $x, y$  into one comparator. On the second layer of the network we split the cube into vertical slices of width 1 in another direction and feed each slice to a  $t^2$ -comparator (see Figure 1c). In other words, for each  $i = 1, \dots, t$  we feed entries  $a_{xiz}$  for all  $x, z$  into one comparator. On the third layer we split the cube into horizontal slices of width 2 (for odd  $t$  the last slice is of width 1) and feed the slices to comparators of arity at most  $2t^2$  (see Figure 1d). Finally, on the fourth layer of the network we split the cube into horizontal slices of width 2 again, but now the first slice is of width 1 (for even  $t$  the last slice is of width 1 as well). Thus, the slices on this layer are shifted compared to the previous one (see Figure 1e).

**Step 2 – correctness.** It remains to prove that this sorting network sorts correctly. Consider any input  $x \in \{0, 1\}^n$ . Note that the cube consists of  $t^2$  vertical columns with  $t$  entries in each column: each column  $A_{yz}$  is obtained by fixing  $y$  and  $z$  in  $a_{xyz}$  and considering all possible  $x$ . We are interested in the weight  $w_{yz}$  of each column, that is the number of 1s in it. For the input  $A$  the weights of the columns can be any numbers from 0 to  $t$ . Now consider the array after the first layer of the network. Note that now each vertical slice of the first layer of the network is sorted. This means that in each of these slices in the first several rows (from bottom to top) there are only 0s, then there might be a row containing both 0s and 1s, and then all remaining rows contain 1s. In particular, the weights of two columns in the same slice differ by at most 1.

Now consider the second layer of the network and consider two different slices  $S_i = \{a_{x,i,z} \mid x, z \in [t]\}$  and  $S_j = \{a_{x,j,z} \mid x, z \in [t]\}$ . Note that each of them contains exactly one column from each slice of the first layer. We know that the weights of the columns in the same slice of the first layer differ by at most 1. Thus, in total, the number of 1s in two slices of the second layer differ by at most  $t$ . In other words, for each  $z$  the first slice contains the column  $A_{iz}$  and the second slice contains the column  $A_{jz}$ . We know that on the input of the second layer of the network  $|w_{iz} - w_{jz}| \leq 1$ . Thus,

$$\left| \sum_z w_{iz} - \sum_z w_{jz} \right| \leq t.$$

Denote by  $r_i$  the number of rows in slice  $S_i$  that consists of only 1s after the second layer of the network. We just showed that the slice  $S_i$  can have one more extra row of 1s, one less row of 1s or something in between. Overall, for the number  $r_j$  of rows consisting of 1s in  $S_j$  we have  $|r_i - r_j| \leq 1$ . As a result, the weights of columns in slices  $S_i$  and  $S_j$  can differ by at most 2. Since this is true for any  $i$  and  $j$ , we have that the weights of all columns in the cube after the second layer of the sorting network differ by at most 2.

To put it another way, there is a horizontal slice of width 2, such that below this slice we have only 0s and above this slice we have only 1s. Thus it remains to sort entries of this slice. Note that on layers 3 and 4 of the network there is a comparator that sorts exactly this slice. Note that by Lemma 4 all other comparators of layers 3 and 4 do not harm the sorting. ◀

This argument can be extended to the cubes of arbitrary dimension  $d$ . More specifically, for  $n = t^d$  and for  $k = (d-1)t^{d-1}$  we can represent entries of an input array as a  $d$ -dimensional cube with side  $d$ , sort “vertical” slices (we need to fix one of the coordinates in  $d$ -dimensional space as vertical) in all  $d - 1$  directions and then sort horizontal slices. This results into

$(d - 1)$  layers of the sorting network and for horizontal slices we need recursive calls for the arrays of size approximately  $2dt^{d-1}$ . Actually, it is expensive to make two recursive calls for horizontal layers, instead we use an additional trick to make just one recursive call.

Although our  $k$ -sorting network construction can be expressed in terms of high dimensional hypercubes, we prefer to give a more general exposition, using a concept of  $s$ -sorted arrays.

## 4.2 Merging $s$ -Sorted Arrays

The following definition plays a key role in our sorting network construction.

► **Definition 16.** *A 0/1-array  $A$  of length  $n$  is  $s$ -sorted if there is an integer interval  $I = \{i, \dots, i + s - 1\} \subseteq [n]$ , such that  $A[j] = 0$  for  $j < i$  and  $A[j] = 1$  for  $j \geq i + s$ . We call  $I$  unsorted interval.*

As an immediate corollary of Lemma 4, we get the following.

► **Corollary 17.** *Suppose a sorting network gets an  $s$ -sorted array with unsorted interval  $I$ . Then the output is also  $s$ -sorted with  $I$  as an unsorted interval.*

We give a construction of a depth-1 sorting network that “merges”  $p$  arrays of length  $n$  that are already  $s$ -sorted into one array which is  $(sp + O(np^2/k))$ -sorted, where  $k$  is the arity of the sorting network.

► **Lemma 18.** *Assume that  $k \geq tp$  for some integers  $t$  and  $p$ . Suppose we have  $p$   $s$ -sorted arrays of size  $n$  each. Assume additionally that  $n$  is divisible by  $t$ . Then there is a depth-1  $k$ -sorting network that merges these arrays into one array of size  $np$  that is  $(sp + 2\frac{np}{t})$ -sorted. If additionally we assume that  $s$  is divisible by  $n/t$ , then the resulting array is  $(sp + \frac{np}{t})$ -sorted.*

**Proof.** Represent each array as a table with  $\frac{n}{t}$  columns and  $t$  rows. We assume the following ordering on the entries of this table: to compare two entries, we first compare the indices of their rows, and then the indices of their columns. Position the tables one under another in a unified table with  $tp$  rows. Note that  $tp \leq k$  and apply  $k$ -comparator to each column in parallel. We claim that the resulting array in the large table is  $(sp + 2\frac{np}{t})$ -sorted.

To see that observe, that in each small table, an unsorted interval of length at most  $s$  occupies at most  $\lceil \frac{st}{n} \rceil + 1$  rows (any other row either consists entirely of 0s or entirely of 1s). In the large table, this gives us at most  $p \left( \lceil \frac{st}{n} \rceil + 1 \right)$  non-constant rows. After sorting each column individually, 0-rows will move to the top, 1-rows will move to the bottom and all other  $p \left( \lceil \frac{st}{n} \rceil + 1 \right)$  rows will be in between on them. They constitute an unsorted interval and the size of it is at most

$$\frac{n}{t} \cdot p \left( \left\lceil \frac{st}{n} \right\rceil + 1 \right).$$

For general  $s$  we can upper bound this as follows

$$\frac{n}{t} \cdot p \left( \left\lceil \frac{st}{n} \right\rceil + 1 \right) \leq \frac{n}{t} \cdot p \left( \frac{st}{n} + 2 \right) = sp + 2\frac{np}{t}.$$

If  $s$  is divisible by  $n/t$ , note that we can just drop the rounding operation and the size of an unsorted interval is at most

$$\frac{n}{t} \cdot p \left( \left\lceil \frac{st}{n} \right\rceil + 1 \right) = \frac{n}{t} \cdot p \left( \frac{st}{n} + 1 \right) = sp + \frac{np}{t}. \quad \blacktriangleleft$$

Applying previous lemma several times we get the following.

► **Lemma 19.** Consider arbitrary  $n$  and  $k$  and denote  $t = \lfloor \sqrt{k} \rfloor$ . Then there is a  $k$ -sorting network of depth  $\lceil \log_t n \rceil - 1$  that on any input outputs an  $s$ -sorted array for  $s \leq \frac{2\lceil \log_t n \rceil n}{t}$ .

**Proof.** Denote  $d = \lceil \log_t n \rceil$  and observe that  $n \leq t^d$ . Introduce the following notation:

$$n_i = \begin{cases} t^{i+1} & \text{for } i = 1, \dots, d-2, \\ t^{d-1}p & \text{for } i = d-1, \end{cases}$$

where  $p$  is such that  $t^{d-1}(p-1) < n \leq t^{d-1}p$ . In particular, since  $p \geq 2$ , we have  $p-1 \geq p/2$  and

$$n > t^{d-1}(p-1) \geq t^{d-1}p/2.$$

For the convenience of presentation, we add  $t^{d-1}p - n$  dummy inputs equal to 1 to the end of the array to make the size of the input to be equal to  $t^{d-1}p$ . By Lemma 4 these inputs will never change their position and can be removed from the sorting network.

We start with an unsorted array as an input. Applying Lemma 18 several times, we get the array consisting of blocks that are  $s$ -sorted for some  $s$ . More specifically, after level  $i$  of the network we get the blocks of size  $n_i$  that are  $s_i$  sorted for

$$s_i = \begin{cases} (i-1)t^i & \text{for } i = 1, \dots, d-2, \\ (d-2)t^{d-2}p & \text{for } i = d-1. \end{cases}$$

On the first step we split the input into blocks of size  $t^2$  and apply comparators to them, the resulting blocks are 0-sorted.

On the  $i$ -th step for  $i = 2, \dots, d-1$  we already have blocks of size  $n_{i-1} = t^i$  from the previous step that are  $s_{i-1}$ -sorted for  $s_{i-1} = (i-2)t^{i-1}$ . Note that  $n_{i-1} = t^i$  is divisible by  $t$  and  $s_{i-1}$  is divisible by  $n_i/t = t^{i-1}$ . We apply Lemma 18 and for  $i < d-1$  get blocks of size  $n_{i-1}t = n_i$  that are  $s$ -sorted for  $s = s_{i-1}t + n_{i-1} = (i-1)t^i$ . For  $i = d-1$  we have just  $p$  subarrays to merge and after the step we get the whole array of size  $t^{d-1}p$  that is  $s$ -sorted for  $s = (d-3)t^{d-2}p + \frac{t^{d-1}p}{t} = (d-2)t^{d-2}p$ .

Finally, observe that

$$s \leq (d-2)t^{d-2}p \leq d \frac{2n}{t}$$

as desired. ◀

### 4.3 Computing Majority

Before constructing a sorting network we solve a simpler task of computing majority function.

► **Theorem 20.** For any  $n$  and for any  $k$  such that  $\log k = \omega(\log \log n)$  (or, to put it differently,  $k$  is growing faster than any  $\text{polylog}(n)$ ), there exists a  $\text{MAJ}_k$ -circuit for  $\text{MAJ}_n$  of depth at most  $(2 + o(1)) \log_k^2 n$ .

The rest of the section is devoted to the proof of Theorem 20.

First observe that to compute  $\text{MAJ}_n$  correctly by a monotone circuit it is enough to compute it correctly on minterm and maxterm inputs: the computation on other inputs follows by monotonicity. Thus, we can assume in our construction that the input contains almost the same number of 0s and 1s. We will construct a sorting network that sorts all such inputs correctly. From the sorting network we get the circuit of the same depth.

## 50:14 Towards Simpler Sorting Networks and Monotone Circuits for Majority

Suppose we need to sort an array of size  $n$  with approximately the same number of 0s and 1s. We apply Lemma 19 to the array. This results in a  $Y$ -sorted array for  $Y = \frac{2\lceil \log_t n \rceil n}{t}$  for  $t = \lfloor \sqrt{k} \rfloor$ . Since the number of 0s and 1s in the array is approximately equal, the smallest  $\frac{n}{2} - Y$  and the largest  $\frac{n}{2} - Y$  elements are sorted correctly (otherwise, the length of the unsorted interval is larger than  $Y$ ). Thus, it remains to sort a specific interval of length  $2Y$  and we can do this recursively.

Overall, we get the following recursive relation.

$$T(n) \leq \lceil \log_t n \rceil - 1 + T(2Y) \leq \log_t n + T\left(\frac{4\lceil \log_t n \rceil n}{t}\right).$$

To solve this recursive relation we use the following lemma.

► **Lemma 21.** *Assume that  $\log k = \omega(\log \log n)$ . Suppose that  $T(n) = \text{const}$  for  $n$  up to some constant and*

$$T(n) \leq 2\log_k n + C + T\left(\left\lceil \frac{D(\log_k n)n}{\sqrt{k}} \right\rceil\right)$$

for some constants  $C$  and  $D > 0$ . Then  $T(n) \leq (2 + o(1))\log_k^2 n$ .

**Proof.** To simplify the presentation, we ignore rounding of the argument of  $T$  first, and address it later. Denote  $\alpha = \frac{\sqrt{k}}{D\log_k n}$ .

We have

$$\begin{aligned} T(n) &\leq 2\log_k n + C + T\left(\frac{n}{\alpha}\right) \\ &\leq 2\log_k n + C + 2\log_k \frac{n}{\alpha} + C + T\left(\frac{n}{\alpha^2}\right) \\ &\leq 2 \sum_{i=0}^{\log_\alpha n} \left(\log_k \frac{n}{\alpha^i} + C\right) \\ &= 2(\log_k n + (\log_k n - \log_k \alpha) + (\log_k n - 2\log_k \alpha) + \dots + 0) + 2C \log_\alpha n \\ &\leq 2 \frac{\log_k n \log_k n}{\log_k \alpha} + 2C \log_\alpha n = \log_k^2 n \log_\alpha k + 2C \log_\alpha n. \end{aligned}$$

It is easy to see that the term  $2C \log_\alpha n$  is negligible, since  $\alpha \gg k^{1/3}$ .

We analyze  $\log_\alpha k$  factor separately:

$$\begin{aligned} \log_\alpha k &= \log_{\frac{\sqrt{k}}{D\log_k n}} k = \frac{\log_2 k}{\log_2 \frac{\sqrt{k}}{D\log_k n}} \leq \frac{\log_2 k}{\frac{1}{2} \log_2 k - D - \log_2 \log_k n} \\ &= \frac{\log_2 k}{\frac{1}{2} \log_2 k - D - \log_2 \log_2 n + \log_2 \log_2 k}. \end{aligned}$$

For  $\log k = \omega(\log \log n)$  this term is  $2 + o(1)$  and we have

$$T(n) \leq (2 + o(1))\log_k^2 n.$$

To address the rounding operation, note that  $\lceil \frac{n}{\alpha} \rceil \leq \frac{n}{\alpha} + 1 \leq \frac{2n}{\alpha}$  for  $\frac{n}{\alpha} \geq 1$ . Thus, in the presence of rounding we will have  $\sum_i \log_k \frac{2n}{\alpha}$  in the calculation above instead of  $\sum_i \log_k \frac{n}{\alpha}$ . This amounts to substituting  $D$  by  $2D$  and does not change the result of the calculation since  $D$  is an arbitrary constant. ◀

## 4.4 Constructing Sorting Network

In this section, we finish the proof of Theorem 2.

We adopt the same strategy as for the computation of majority. More specifically, we apply Lemma 19 recursively to get  $s$ -sorted array for smaller and smaller  $s$ . However, now our task is more tricky. In the proof of Theorem 20 when we get to an  $s$ -sorted array we know exactly where the unsorted interval is located (in the middle of the array). However, now we need to sort arbitrary input arrays and an unsorted interval can be anywhere.

We construct the network recursively. We assume that at the beginning of each step, we have an  $s$ -sorted array (at the beginning of the process  $s = n$ ). Denote the unsorted interval by  $A$ ,  $|A| \leq s$ . Split the array into consecutive blocks  $B_1, \dots, B_p$  of size  $s$  (the last block  $B_p$ ) might be smaller.

The recursive step consists of two stages. In the first stage, we split the array into blocks  $B_1 \cup B_2, B_3, \cup B_4$ , and so on, each block of size  $2s$  (one last block might be smaller). In the second stage, we split the array into blocks  $B_1, B_2 \cup B_3, B_4 \cup B_5$ , and so on (again the last block might be smaller than  $2s$ ). Before describing each of the stages, observe that either in the first stage or in the second stage (or in both) the interval  $A$  falls completely into one of the blocks. Indeed,  $A$  can intersect with at most two consecutive blocks  $B_i, B_{i+1}$  and in one of the stages, they form a single block.

In the first stage, we apply Lemma 19 to each of the blocks  $B_1 \cup B_2, B_3, \cup B_4, \dots$  separately. As a result, each block is  $s'$ -sorted for  $s' \leq \frac{4 \lceil \log_{\lfloor \sqrt{k} \rfloor} n \rceil s}{\lfloor \sqrt{k} \rfloor}$ . Moreover, if the block consisted of only 0s and 1s, then it does not change.

If  $A$  is contained in one of the blocks of the first stage, we are already done: there is only one initially unsorted block that by Lemma 19 after the step is  $s'$ -sorted. By Corollary 17 this property remains true after the additional comparators we apply for the other case.

If  $A$  is split between two blocks of the first stage, then after the stage we have two consecutive unsorted blocks, each of them is  $s'$ -sorted. Denote unsorted parts by  $C_1, C_2$ . Note that by Corollary 17  $C_1, C_2 \subseteq A$  and thus,  $C_1$  and  $C_2$  fall into one block of the second stage. It is tempting to apply Lemma 19 to the blocks of the second stage as well. However, this application is too expensive and will not result in the desired bound.

Instead we do the following. We represent each block of the second stage (of size at most  $2s$ ) as a table with  $p = \lceil 2s/k \rceil$  columns and  $k$  rows, filled in row by row from top to bottom. For convenience, if the last row is not complete, add dummy variables equal to 1 to complete the row.

Each of the intervals  $C_1, C_2$  occupy at most  $\lceil s'/p \rceil + 1$  rows. There might be another row that contain a switch between blocks  $B_i$  and  $B_{i+1}$ . Each other row consist either entirely of 0s, or entirely of 1s. Denote the number of all 0 rows by  $a$  and the number of all 1 rows by  $b$ .

We apply a comparator to each column separately. As a result, each column will contain  $a$  zeros in the beginning,  $b$  ones in the end and some part in between. The number of rows in the middle part is at most  $2 \lceil s'/p \rceil + 3$ . The number of entries in these rows is at most

$$s'' = p(2 \lceil s'/p \rceil + 3) \leq 2s' + 5p \leq 3s'$$

for large enough input size. Thus, after the second stage we get  $s''$ -sorted array and we are done with the recursion step.

Thus, we get that  $s'' \leq 12 \frac{\lceil \log_t n \rceil s}{t}$  and we get the following recursive relation

$$T(n) \leq \log_t n + T\left(\frac{12 \lceil \log_t n \rceil n}{t}\right).$$

We apply Lemma 21 again to get  $T(n) \leq (2 + o(1)) \log_k^2 n$ .

This finishes the proof of Theorem 2.

## 4.5 Other Applications

In this section we give two more examples of results that follow from our construction.

► **Lemma 22.** *There is a MAJ<sub>k</sub>-circuit of depth 4 computing MAJ<sub>n</sub> for  $k = O(n^{3/5})$ .*

**Proof.** Denote  $r = \lceil n^{1/5} \rceil$ . For simplicity we pad the input with constants 0 and 1 to make the size of the array  $r^5$  without changing the output of majority. We will use  $k$ -sorters for  $k = 4r^3$ .

As in the proof of Theorem 20 it is enough to compute MAJ<sub>n</sub> on minterms and maxterms, thus we can assume that there are approximately equal number of 0s and 1s in the input.

We will build a  $k$ -sorting network and the existence of the circuit follows. On the first layer of the network we split the input into blocks of size  $r^3$  and sort them. On the second layer we use Lemma 18 with  $p = r$  and  $t = r^2$ . As a result we get blocks of size  $r^4$  that are  $r^2$ -sorted. On the third level we apply Lemma 18 again with the same values of  $p$  and  $t$ . As a result we have that the whole input is now  $2r^3$ -sorted. On the last layer of the network just as in the proof of Theorem 20 we apply  $4r^3$ -comparator to the middle of the array. ◀

In [16] Knuth posed a problem of constructing a minimal depth  $k$ -sorting network for the input of size  $k^2$ . Parker and Parbery [23] gave a construction of depth 9. Here we slightly improve on this at the cost of using comparators of size  $O(k)$ .

► **Lemma 23.** *There is a  $k$ -sorting network of depth 8 that sorts an array of size  $n$  with  $k = O(n^{1/2})$ .*

**Proof.** As usual pad an array with constants to make  $n = r^4$  for some integer  $r$ . Thus  $k = O(r^2)$ .

We follow the same strategy as in Section 4.4. First we apply Lemma 19 that uses three layers of network and results in a  $s$ -sorted array for  $s = O(r^3)$ . Then, we apply Lemma 19 again to the blocks of size  $O(r^3)$  to get a network of depth 2 that results in each block being  $O(r^2)$ -sorted. Then we apply one more layer to merge unsorted intervals in different blocks to get the array that is  $O(r^2)$ -sorted. Finally, we again split the array into blocks, this time of size  $O(r^2)$  to complete the sorting using two layers. In total we use  $3 + 2 + 1 + 2 = 8$  layers. ◀

## 5 Conclusion

The obvious open problems are to come up with explicit constructions of sorting networks and monotone circuits for majority of smaller depth. One specific problem is to extend our  $O(\log^{5/3} n)$  construction to sorting networks. The obstacle that we encountered is that there is no randomized construction of a low-depth sorting network that we can use as a start. Another interesting question is to extend our  $O(\log^{5/3} n)$  construction to get a MAJ<sub>k</sub>-circuit for MAJ<sub>n</sub> of depth  $O(\log_k^{5/3} n)$ . Such a construction can be used instead of  $O(\log_k^2 n)$ -depth circuit in downward self-reduction to further improve the upper bound. Again, the obvious obstacle is that it is not clear how to get a starting construction.

---

### References

- 1 Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in  $c \log n$  parallel steps. *Comb.*, 3(1):1–19, 1983. doi:10.1007/BF02579338.



- 2 Kenneth E. Batchier. Sorting networks and their applications. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*, volume 32 of *AFIPS Conference Proceedings*, pages 307–314. Thomson Book Company, Washington D.C., 1968. doi:10.1145/1468075.1468121.
- 3 Richard Beigel and John Gill. Sorting  $n$  objects with a  $k$ -sorter. *IEEE Trans. Computers*, 39(5):714–716, 1990. doi:10.1109/12.53587.
- 4 Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 276–287. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365687.
- 5 Gianfranco Bilardi and Franco P. Preparata. A minimum area VLSI network for  $o(\log n)$  time sorting. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 64–70. ACM, 1984. doi:10.1145/800057.808666.
- 6 V. Chvátal. Lecture notes on the new AKS sorting network. Technical report, Department of Computer Science, Rutgers University, 1992.
- 7 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 Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013 – 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 185–202. Springer, 2013. doi:10.1007/978-3-642-40084-1\_11.
- 8 Robert Cypher and Jorge L. C. Sanz. Cubesort: A parallel algorithm for sorting  $N$  data items with  $s$ -sorters. *J. Algorithms*, 13(2):211–234, 1992. doi:10.1016/0196-6774(92)90016-6.
- 9 Natalia Dobrokhotova-Maikova, Alexander Kozachinskiy, and Vladimir V. Podolskii. Constant-depth sorting networks. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPICs*, pages 43:1–43:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ITCS.2023.43.
- 10 Caxton C Foster and Fred D Stockton. Counting responders in an associative memory. *IEEE Transactions on Computers*, 100(12):1580–1583, 1971.
- 11 Qingshi Gao and Zhiyong Liu. Sloping-and-shaking. *Science in China Series E: Technological Sciences*, 40(3):225–234, 1997.
- 12 Oded Goldreich. On (valiant’s) polynomial-size monotone formula for majority. In *Computational Complexity and Property Testing: On the Interplay Between Randomness and Computation*, pages 17–23. Springer, 2020.
- 13 Shlomo Hoory, Avner Magen, and Toniann Pitassi. Monotone circuits for the majority function. In *Proceedings of the 9th international conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th international conference on Randomization and Computation*, pages 410–425, 2006.
- 14 Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.
- 15 Nabil Kahalé, Frank Thomson Leighton, Yuan Ma, C. Greg Plaxton, Torsten Suel, and Endre Szemerédi. Lower bounds for sorting networks. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 437–446. ACM, 1995. doi:10.1145/225058.225178.
- 16 Donald Ervin Knuth. *The art of computer programming, Volume III, 2nd Edition*. Addison-Wesley, 1998. URL: <https://www.worldcat.org/oclc/312994415>.
- 17 Clemens Lautemann. Bpp and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

- 18 De-Lei Lee and Kenneth E. Batchier. A multiway merge sorting network. *IEEE Trans. Parallel Distributed Syst.*, 6(2):211–215, 1995. doi:10.1109/71.342136.
- 19 Frank Thomson Leighton. Tight bounds on the complexity of parallel sorting. *IEEE Trans. Computers*, 34(4):344–354, 1985. doi:10.1109/TC.1985.5009385.
- 20 David E Muller and Franco P Preparata. Bounds to complexities of networks for sorting and for switching. *Journal of the ACM (JACM)*, 22(2):195–201, 1975.
- 21 Toshio Nakatani, Shing-Tsaan Huang, Bruce W. Arden, and Satish K. Tripathi. K-way bitonic sort. *IEEE Trans. Computers*, 38(2):283–288, 1989. doi:10.1109/12.16506.
- 22 Ian Parberry. The pairwise sorting network. *Parallel Process. Lett.*, 2:205–211, 1992. doi:10.1142/S0129626492000337.
- 23 Bruce Parker and Ian Parberry. Constructing sorting networks from k-sorters. *Inf. Process. Lett.*, 33(3):157–162, 1989. doi:10.1016/0020-0190(89)90196-8.
- 24 Michael S Paterson. Improved sorting networks with  $o(\log n)$  depth. *Algorithmica*, 5(1):75–92, 1990.
- 25 Claus-Peter Schnorr and Adi Shamir. An optimal sorting algorithm for mesh connected computers. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 255–263. ACM, 1986. doi:10.1145/12130.12156.
- 26 Joel Seiferas. Sorting networks of logarithmic depth, further simplified. *Algorithmica*, 53(3):374–384, 2009.
- 27 Sandeep Sen, Isaac D. Scherson, and Adi Shamir. Shear sort: A true two-dimensional sorting techniques for VLSI networks. In *International Conference on Parallel Processing, ICPP'86, University Park, PA, USA, August 1986*, pages 903–908. IEEE Computer Society Press, 1986.
- 28 Feng Shi, Zhiyuan Yan, and Meghanad D. Wagh. An enhanced multiway sorting network based on n-sorters. In *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014, Atlanta, GA, USA, December 3-5, 2014*, pages 60–64. IEEE, 2014. doi:10.1109/GlobalSIP.2014.7032078.
- 29 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 330–335, 1983.
- 30 S. S. Tseng and Richard C. T. Lee. A parallel sorting scheme whose basic operation sorts  $N$  elements. *Int. J. Parallel Program.*, 14(6):455–467, 1985. doi:10.1007/BF00991185.
- 31 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.
- 32 Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.
- 33 Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18:337–375, 2009.
- 34 Andrew Chi-Chih Yao. Bounds on selection networks. *SIAM J. Comput.*, 9(3):566–582, 1980. doi:10.1137/0209043.
- 35 Lijun Zhao, Zhiyong Liu, and Qingshi Gao. An efficient multiway merging algorithm. *Science in China Series E: Technological Sciences*, 41(5):543–551, 1998.

# Consequences of Randomized Reductions from SAT to Time-Bounded Kolmogorov Complexity

Halley Goldberg ✉

Simon Fraser University, Burnaby, Canada

Valentine Kabanets ✉

Simon Fraser University, Burnaby, Canada

---

## Abstract

A central open question within meta-complexity is that of NP-hardness of problems such as MCSP and  $MK^tP$ . Despite a large body of work giving consequences of and barriers for NP-hardness of these problems under (restricted) deterministic reductions, very little is known in the setting of randomized reductions. In this work, we give consequences of randomized NP-hardness reductions for both approximating and exactly computing time-bounded and time-unbounded Kolmogorov complexity.

In the setting of *approximate*  $K^{\text{poly}}$  complexity, our results are as follows.

1. Under a derandomization assumption, for any constant  $\delta > 0$ , if approximating  $K^t$  complexity within  $n^\delta$  additive error is hard for SAT under an honest randomized non-adaptive Turing reduction running in time polynomially less than  $t$ , then  $\text{NP} = \text{coNP}$ .
2. Under the same assumptions, the worst-case hardness of NP is equivalent to the existence of one-way functions.

Item 1 above may be compared with a recent work of Saks and Santhanam [39], which makes the same assumptions except with  $\omega(\log n)$  additive error, obtaining the conclusion  $\text{NE} = \text{coNE}$ .

In the setting of *exact*  $K^{\text{poly}}$  complexity, where the barriers of Item 1 and [39] do not apply, we show:

3. If computing  $K^t$  complexity is hard for SAT under reductions as in Item 1, then the average-case hardness of NP is equivalent to the existence of one-way functions. That is, “Pessiland” is excluded.

Finally, we give consequences of NP-hardness of *exact time-unbounded* Kolmogorov complexity under randomized reductions.

4. If computing Kolmogorov complexity is hard for SAT under a randomized many-one reduction running in time  $t_R$  and with failure probability at most  $1/(t_R)^{16}$ , then  $\text{coNP}$  is contained in non-interactive statistical zero-knowledge; thus  $\text{NP} \subseteq \text{coAM}$ . Also, the worst-case hardness of NP is equivalent to the existence of one-way functions.

We further exploit the connection to NISZK along with a previous work of Allender et al. [7] to show that hardness of K complexity under randomized many-one reductions is highly robust with respect to failure probability, approximation error, output length, and threshold parameter.

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

**Keywords and phrases** Meta-complexity, Randomized reductions, NP-hardness, Worst-case complexity, Time-bounded Kolmogorov complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.51

**Category** RANDOM

**Related Version** *Full Version:* <https://eccc.weizmann.ac.il/report/2024/120/> [15]

**Funding** *Halley Goldberg:* Supported by NSERC CGS D.

*Valentine Kabanets:* Supported by NSERC Discovery research grant.



© Halley Goldberg and Valentine Kabanets;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 51; pp. 51:1–51:19



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Meta-complexity aims to determine the computational complexity of the tasks to compute various intrinsic complexity measures of given binary strings. Two prominent examples of such complexity measures are the minimum circuit size of a given truth table of a Boolean function, and the minimum time-bounded Kolmogorov complexity (denoted  $K^t$ ) of a given binary string. The corresponding meta-complexity problems are the *Minimum Circuit Size Problem* (MCSP):

given a binary string  $x \in \{0, 1\}^{2^n}$  and a parameter  $s \leq 2^n$ , decide if there is an  $n$ -input boolean circuit of size at most  $s$  whose truth table equals  $x$ ,

and the *Minimum  $K^t$  Problem* ( $MK^tP$ ):

given a binary string  $x \in \{0, 1\}^n$ , and a parameter  $s \leq n$ , decide if there is a binary input  $w$  of length at most  $s$  such that some fixed universal Turing machine  $U$  on input  $w$  prints  $x$  within  $t$  time steps.

The history of these two problems goes back to at least the 1950s and '60s. In the Soviet Union, during that period, those involved in “theoretical cybernetics” were keenly interested in problems related to switching circuits and Kolmogorov’s new theory of the complexity of strings. It was widely suspected that one could not avoid *perebor* (exhaustive search) in the solution of the corresponding minimization problems. Levin’s interest in *perebor*, culminating in his discovery of NP-completeness in the early 1970s, was motivated in particular by questions about the complexity of time-bounded Kolmogorov complexity [40]. Since then, both MCSP and  $MK^tP$  have resisted categorization as efficiently decidable or as NP-complete, a somewhat uncommon state of affairs for natural problems in NP.

In 2000, Kabanets and Cai took up the study of circuit minimization again, with a result suggesting that NP-hardness of MCSP may be very difficult to resolve: if MCSP is NP-hard under a deterministic many-one reduction such that output length depends only on input length, then one gets the lower bound  $E \not\subseteq P/\text{poly}$  [32]. At least, if MCSP is NP-hard, then showing its hardness would seem to require different techniques than those applied in the past, barring any further major breakthroughs. A line of work has continued to push further in this negative direction, progressively obtaining (1) “stronger” consequences, and (2) consequences of NP-hardness under more powerful forms of reducibility. An example of the former is a result of Murray and Williams, which obtains  $NP \not\subseteq P/\text{poly}$  from NP-hardness of MCSP under log-time uniform  $AC^0$  reductions [35]. An example of the latter is a result of Hitchcock and Pavan, which obtains  $EXP \neq ZPP$  from NP-hardness of MCSP under deterministic non-adaptive Turing reductions [27]. There are many more examples of this kind of work relying essentially on the determinism of the reductions in question; see, e.g., [6, 38, 8, 26].<sup>1</sup>

In contrast to the negative line of work for deterministic reductions, there is a positive line of work obtaining NP-hardness of variants of MCSP and  $MK^tP$  that seem to come progressively closer to the standard definitions of these problems. Examples include [29, 23, 30, 28]. A common feature of these results is their employment of randomness in the NP-hardness reductions. An impressive example of such a result is Hirahara’s recent proof of NP-hardness of partial-function versions of MCSP and  $MK^tP$  [23]. Additionally, from [5], MCSP is hard

<sup>1</sup> One result of [26] deals with one-query randomized reductions to  $MCSP^{\mathcal{O}}$  working for *every* oracle  $\mathcal{O}$ , which may be seen as an exception. Other results of that work give consequences of deterministic reductions to, for example, approximating circuit size and Levin’s  $K^t$  complexity.

for SZK (statistical zero-knowledge) under randomized reductions, which is the strongest unconditional hardness known for MCSP. All of this begs the question whether randomness is the key ingredient for the hardness of problems in meta-complexity: most barriers apply to deterministic reductions, whereas most progress has been made via randomized reductions.

As for the negative direction for randomized reductions, there has been far less headway. In fact, prior to this work, only two such results were known for MCSP and  $\text{MK}^t\text{P}$ . Murray and Williams ruled out NP-hardness of MCSP in the very restrictive setting of poly-logarithmic-time randomized projections [35]. More recently, Saks and Santhanam showed that  $\text{NE} = \text{coNE}$  if approximating  $\text{K}^t$ -complexity is NP-hard under randomized non-adaptive polynomial-time reductions (with some caveats, including a derandomization assumption and that the time-bound  $t$  in the superscript of  $\text{K}^t$  must be greater than the running time of the reduction) [39].

Of course, any NP-hardness of  $\text{MK}^t\text{P}$  or MCSP would be a major breakthrough for complexity theory, including hardness under a non-black-box reduction. In that sense, the *kind* of reduction in question is hardly important in itself. That being said, obtaining consequences of restricted forms of reduction can certainly help guide the “search for NP-hardness”. For example, a recent work of Ilango proved that approximating  $\text{K}^t$  within  $\Omega(n)$  additive error is NP-hard in the random oracle model [29]. As mentioned in that paper, the reduction circumvents the barrier of [39] by requiring more time than the superscript  $t$ . As with much of complexity theory, one can always take negative results as putting into focus the space for positive progress.

In this paper, we advance in the negative direction for randomized reductions, obtaining results with stronger consequences and from reductions to harder problems compared to prior work.

## 2 Main Results

We show a number of consequences of the assumptions that there exist restricted randomized NP-hardness reductions for the exact and approximate variants of the problem to determine the (time-bounded) Kolmogorov complexity of a given binary string.

In addition to the problem  $\text{MK}^t\text{P}$  introduced above, we shall also consider its time-unbounded version, MKP, where given a binary string  $x \in \{0, 1\}^n$  and a threshold parameter  $s \leq n$ , one needs to decide if there is a string  $w \in \{0, 1\}^{\leq s}$  such that a fixed universal TM  $U(w)$  outputs  $x$ . We also consider the probabilistic variant of  $\text{K}^t$ , denoted by  $\text{pK}^t$ , where  $\text{pK}^t(x)$  is defined as the minimum length  $s$  such that, for each of at least  $2/3$  of random strings  $r$ , there exists some input  $w_r \in \{0, 1\}^{\leq s}$  such that  $U(w_r, r)$  outputs  $x$  within  $t$  time steps. The corresponding Minimum  $\text{pK}^t$  Problem is denoted by  $\text{MpK}^t\text{P}$ . For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $\mu \in \{\text{K}, \text{pK}\}$ ,  $\text{Approx}_g\text{-}\mu^t$  refers to the problem of approximating  $\mu^t$  complexity of a given  $x \in \{0, 1\}^n$  to within a  $g(n)$  additive error.  $\text{Approx}_g\text{-K}[s]$  refers to the problem of approximating K complexity except with threshold parameter fixed to  $s$ .

### 2.1 Consequences of showing the NP-hardness of an approximation to $\text{pK}^t$ or $\text{K}^t$

Informally, our first results show that NP-hardness of  $\text{Approx}_{n^s}\text{-pK}^t$  under honest non-adaptive randomized reductions with runtime sufficiently smaller than  $t$  implies that

- $\text{NP} \subseteq \text{coAM}$  (and hence, the polynomial-time hierarchy collapses [12]), and
- if, in addition, no one-way functions exist, then  $\text{NP} \subseteq \text{BPP}$ ;

here “honest” reductions are those that make queries of length at least some polynomial of the input to the reduction. We also get a similar result for  $\text{Approx}_{n^\delta}\text{-K}^t$ , under a derandomization assumption that  $\text{E}$  requires exponential-size nondeterministic circuits.

More precisely, we show that under the same NP-hardness assumptions, there is a black-box non-adaptive reduction from SAT to inverting an auxiliary input one-way function.<sup>2</sup> Moreover, this reduction is of a restricted form in which the oracle only needs to invert the function on auxiliary input  $\varphi$ , where  $\varphi$  is the input to SAT; this is called a “fixed-auxiliary-input reduction” [9]. The “ $\gamma$ -honesty” condition below means that all queries  $q \in \{0,1\}^*$  made by the reduction are such that  $|q| \geq n^\gamma$ , where  $n$  is the length of the input to the reduction.

► **Theorem 1** (Collapsing the Polynomial Hierarchy). *For any constants  $\delta, \gamma > 0$ , there is a polynomial  $p$  such that, for any  $t, t_R : \mathbb{N} \rightarrow \mathbb{N}$  satisfying  $p(t_R(n)) \leq t(n)$  for all  $n \in \mathbb{N}$ , we have the following.*

1. *If  $\text{Approx}_{n^\delta}\text{-pK}^t$  is hard for SAT under a  $\gamma$ -honest non-adaptive randomized reduction running in time  $t_R$ , then there is a black-box non-adaptive fixed-auxiliary-input reduction from SAT to inverting an auxiliary-input OWF. The latter implies that*

$$\text{NP} \subseteq \text{coAM}.$$

2. *Assume  $\text{E} \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ . If  $\text{Approx}_{n^\delta}\text{-K}^t$  is hard for SAT under an honest non-adaptive randomized reduction running in time  $t_R$ , then*

$$\text{NP} = \text{coNP}.$$

As a consequence of the above non-adaptive black-box reduction from SAT to inverting an auxiliary-input one-way function, we further obtain from the hypothesis of Theorem 1 that the existence of a standard one-way function can be based on the worst-case hardness of NP. That is, proving NP-hardness of  $\text{Approx}_{n^\delta}\text{-K}^t$  (under restricted randomized reductions) is as hard as achieving the “holy grail of cryptography”.

We obtain both adaptive black-box and non-adaptive BPP-black-box<sup>3</sup> reductions from SAT to the problem of inverting a standard OWF. The former follows immediately from our Theorem 1 and a recent work of Nanashima [36], and the latter is implicit in [24], though we provide a short, self-contained proof building on Theorem 1.

► **Theorem 2** (Excluding Pessiland and Heuristica). *For any constants  $\delta, \gamma > 0$ , there is a polynomial  $p$  such that, for any  $t_R, t : \mathbb{N} \rightarrow \mathbb{N}$  satisfying  $p(t_R(n)) \leq t(n)$  for all  $n \in \mathbb{N}$ , we have the following.*

1. *If  $\text{Approx}_{n^\delta}\text{-pK}^t$  is hard for SAT under a  $\gamma$ -honest non-adaptive randomized reduction running in time  $t_R$ , then there exist both (I) a black-box adaptive randomized polynomial-time reduction, and (II) a BPP-black-box non-adaptive randomized polynomial-time reduction, from SAT to inverting a OWF. As a consequence, we get*

$$\text{NP} \not\subseteq \text{BPP} \iff \exists \text{ OWF}.$$

2. *Assume  $\text{E} \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ . If  $\text{Approx}_{n^\delta}\text{-K}^t$  is hard for SAT under an honest non-adaptive randomized reduction running in time  $t_R$ , then*

$$\text{NP} \neq \text{P} \iff \exists \text{ OWF}.$$

<sup>2</sup> We consider auxiliary input functions  $f = \{f_\varphi\}_{\varphi \in \{0,1\}^*}$  as defined in [37].

<sup>3</sup> As defined by [18], a BPP-black-box reduction  $R$  from a problem  $L$  to a problem  $L'$  is an efficient oracle Turing machine that correctly decides  $L$ , given any oracle  $A \in \text{BPP}$  such that  $A$  decides  $L'$ .

With a similar argument, we also get the following statement for Levin's  $K^t$  complexity.

► **Corollary 3.** *For any constant  $\delta > 0$ , we have the following. Assume  $E \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ . If  $\text{Approx}_{n^\delta}\text{-K}^t$  is hard for SAT under an honest non-adaptive randomized reduction, then  $\text{NP} = \text{coNP}$ . Moreover, if no one-way functions exist, then  $\text{NP} = \text{P}$ .*

## 2.2 Consequences of showing the NP-hardness of $K^t$

Though the conclusions of Theorems 1 and 2 are incomparable, one may find  $\text{NP} \subseteq \text{coAM}$  unbelievable, in which case Theorem 2 would not appear to yield a promising route for actually excluding Pessiland and Heuristica. Indeed, the earlier barrier result of [39] was part of Hirahara's motivation to introduce a harder "distributional" variant of  $K^t$  complexity in a recent work [24], delineating an intact positive approach for excluding Impagliazzo's worlds via NP-hardness of meta-complexity.

As a counterpoint, building on a work of Liu and Pass [33], we show that NP-hardness of *exact*  $K^t$  complexity would still suffice to exclude Pessiland while circumventing the barrier of Theorem 1 (and [39]). As noted in [33], problems of exact and approximate  $K^t$  complexity are qualitatively different: approximating  $K^t$  within  $\omega(\log n)$  additive error is unconditionally easy on average (in the "error-prone" sense) over the uniform distribution, but the argument fails in the setting of exact  $K^t$ . Thus, there is still room for optimism with regard to excluding Pessiland via NP-hardness of standard  $K^t$  complexity.

► **Theorem 4** (Excluding Pessiland). *There is a polynomial  $p$  such that, for any  $t, t_R : \mathbb{N} \rightarrow \mathbb{N}$  satisfying  $t(n) \geq p(t_R(n))$  for all  $n \in \mathbb{N}$ , we have the following.*

1. *If  $\text{Mpk}^t\text{P}$  is hard for SAT under an honest non-adaptive randomized reduction running in time  $t_R$ , then there is a black-box average-case reduction from SAT to inverting OWFs. As a consequence, we get that*

$$\text{DistNP} \not\subseteq \text{HeurBPP} \iff \exists \text{OWF}.$$

2. *Assume  $E \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ . If  $\text{MK}^t\text{P}$  is hard for SAT under an honest non-adaptive randomized reduction running in time  $t_R$ , then*

$$\text{DistNP} \not\subseteq \text{HeurP} \iff \exists \text{OWF}.$$

## 2.3 Consequences of showing the NP-hardness of $K$

Finally, we show that NP-hardness of Kolmogorov complexity under randomized many-one reductions would imply  $\text{NP} \subseteq \text{coAM}$  and a collapse of the polynomial hierarchy. To the best of our knowledge, this is the first evidence against NP-hardness of exact Kolmogorov complexity under randomized many-one reductions. We also get under the same assumption that if  $\text{NP} \not\subseteq \text{BPP}$  then one-way functions exist.

► **Theorem 5** (Collapsing the Polynomial Hierarchy). *There is a polynomial  $p$  such that, for any  $t_R : \mathbb{N} \rightarrow \mathbb{N}$ , we have the following. If MKP is hard for SAT under a randomized polynomial time many-one reduction running in time  $t_R(n)$  and with failure probability at most  $1/p(t_R(n))$ , then*

$$\text{NP} \subseteq \text{coAM}.$$

*If, in addition, no one-way functions exist, then  $\text{NP} \subseteq \text{BPP}$ .*

## 2.4 Robustness of reductions to K

In fact, we can get a stronger result than that stated above: namely, we show that if a decidable language  $L$  reduces to MKP as in Theorem 5, then  $\bar{L} \subseteq \text{NISZK}$ , where NISZK is the class of promise problems admitting *non-interactive* statistical zero-knowledge proofs. In particular, we prove the following.

► **Theorem 6.** *For any polynomial  $t_R$  and decidable language  $L$ , if MKP is hard for  $L$  under a randomized many-one reduction running in time  $t_R(n)$  and with failure probability at most  $1/t_R(n)^{16}$ , then  $\bar{L} \subseteq \text{NISZK}$ .*

Since it is known that  $\text{NISZK} \subseteq \text{SZK} \subseteq \text{AM} \cap \text{coAM}$  [17, 14, 1], where SZK is the class of problems admitting statistical zero-knowledge proofs, Theorem 6 captures Theorem 5. It also improves on the following statement implicit in a previous work of Allender et al. [7].

► **Theorem 7 ([7]).** *For any decidable language  $L$ , if  $\text{Approx}_{\omega(\log n)}\text{-K}[n/2]$  is hard for  $L$  under an honest randomized many-one reduction with failure probability at most  $1/n^{\omega(1)}$ , then  $\bar{L} \subseteq \text{NISZK}$ .*

Note that Theorem 6 improves on Theorem 7 in three respects: we do not require the reduction to be honest, we do not require an  $\omega(\log n)$  approximation term, and we do not require the threshold parameter to be fixed.

Combining the above with a converse provided in [7], we show that hardness of MKP under randomized many-one reductions (with sufficiently small failure probability) is remarkably robust with respect to approximation error, failure probability, honesty, and threshold parameter (fixed or unfixed). For instance, if MKP is NP-hard under a  $t_R(n)$ -time many-one reduction with failure probability  $1/\text{poly}(t_R(n))$ , then it is also NP-hard under a polynomial-time many-one reduction with exponentially small failure probability. More specifically,

► **Theorem 8.** *There is a polynomial  $p$  such that for any decidable language  $L$  and polynomial  $t_R$ , the following are equivalent.*

1.  $\bar{L} \subseteq \text{NISZK}$ ;
2. MKP is hard for  $L$  under a randomized many-one reduction running in time  $t_R(n)$  and with two-sided failure probability at most  $1/p(t_R(n))$ ;
3.  $\text{Approx}_{n^{o(1)}}\text{-K}[n/2]$  is hard for  $L$  under an honest randomized many-one reduction with one-sided failure probability at most  $2^{-\text{poly}(n)}$ .

## 3 Related Work

Saks and Santhanam obtain a barrier result similar to our Theorem 1, Item 2, for the regime of super-logarithmic additive error. Specifically, they prove the following.

► **Theorem 9 ([39]).** *Assume  $\text{E} \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ . There is a polynomial  $p$  satisfying the following. For any  $t, t_R : \mathbb{N} \rightarrow \mathbb{N}$  such that  $p(t_R(n)) \leq t(n)$ , if  $\text{Approx}_{\omega(\log n)}\text{-K}^t$  is hard for SAT under an honest, fixed query length, non-adaptive randomized reduction running in time  $t_R$ , then  $\text{NE} = \text{coNE}$ .*

Here, “fixed query length” means that the lengths of all queries made in the reduction are identical and depend only on the length of the input to the reduction, independent of randomness. In comparison, at the cost of increasing the approximation error term from  $\omega(\log n)$  to  $n^\delta$  for any constant  $\delta > 0$ , we obtain the stronger (and presumably less believable) consequence  $\text{NP} = \text{coNP}$ . Moreover, we do not require that the reduction have fixed query



length: in our case, the length of queries need not be the same, and they can depend on the input and the randomness of the reduction. The honesty condition is identical in this work and [39]. We also note that our proof techniques can be made to capture the regime of  $\omega(\log n)$  additive error, in which case we recover the statement of [39] improved to reductions without fixed query length.

Our Theorem 2 is related to a recent work of Hirahara [24], which introduces a “distributional” variant of  $K^t$  complexity, denoted  $dK^t$ , defined as follows: for a string  $x \in \{0, 1\}^*$ , a time bound  $t \in \mathbb{N}$ , and a distribution  $\mathcal{D}$ ,

$$dK^t(x \mid \mathcal{D}) = \min_{s \in \mathbb{N}} \left\{ \exists d \in \{0, 1\}^s \mid \Pr_{r \sim \mathcal{D}} [U(d, r) \text{ halts and outputs } x \text{ within } t \text{ steps}] \geq 2/3 \right\}.$$

Using the techniques of that work, it is possible to recover a part of our Theorem 2 exactly: namely, the existence of a BPP-black-box non-adaptive reduction from SAT to inverting a OWF. This is essentially due to the fact that if, for example, approximating  $K^t$  is NP-hard, then approximating  $dK^t$  is also NP-hard, since  $dK^t$  captures  $K^t$  when the provided distribution  $\mathcal{D}$  always outputs the empty string. A probabilistic variant of  $dK^t$  is also introduced in [24], which similarly generalizes  $pK^t$ .

However, our proof of Theorem 2 takes a partly different approach to that implicit in [24]. In particular, though both our proof and that work employ a non-black-box worst-case to average-case reduction as in [19, 20, 16], the latter approach would use this kind of reduction in two places: once to reduce NP to inverting an auxiliary-input one-way function, and once to obtain  $\text{NP} \not\subseteq \text{BPP} \implies \text{DistNP} \not\subseteq \text{AvgBPP}$ . To accommodate the reduction to inverting an auxiliary-input OWF, Hirahara introduces a new kind of mildly black-box reduction, which is more restrictive than the standard notion of a class-specific black-box reduction [18]. In contrast, as an intermediate step, we obtain a completely black-box non-adaptive reduction from NP to inverting an auxiliary-input OWF. We employ a class-specific worst-to-average reduction only to obtain  $\text{NP} \not\subseteq \text{BPP} \implies \text{DistNP} \not\subseteq \text{AvgBPP}$ .

As noted above, we could alternatively simply combine our Theorem 1 with [36] to obtain the statement

$$\text{NP} \not\subseteq \text{BPP} \implies \exists \text{OWF}.$$

However, we provide in [15] a self-contained proof of a BPP-black-box non-adaptive reduction. This is for completeness and to clarify the connection to Theorem 1.

Finally, we mention a few previous works related to our Theorem 5. Interestingly, by Allender et al., computing Kolmogorov complexity is known to be hard for PSPACE under deterministic adaptive Turing reductions [4]. This was improved by Hirahara to show that Kolmogorov complexity is hard for  $\text{EXP}^{\text{NP}}$  under deterministic adaptive Turing reductions and hard for NEXP under randomized non-adaptive reductions [21]. Thus, Theorem 5 indicates a sharp contrast between the power of randomized many-one reductions and more powerful reductions with respect to the hardness of Kolmogorov complexity. Saks and Santhanam also prove that NP-hardness of *approximating* Kolmogorov complexity within  $\omega(\log n)$  additive error under honest randomized non-adaptive reductions would imply  $\text{NP} \subseteq \text{coAM}$  [39]. Note that Theorem 5 does not assume honesty.

## 4 Techniques

In this section, we give an overview of the techniques used to prove our main results. Formal details can be found in the full version of the paper [15].

#### 4.1 Proof sketch of Theorem 1

As a warm-up, first consider the case of a deterministic length-increasing many-one reduction. In particular, let  $R$  be such a reduction from SAT to  $\text{Approx}_{n^\delta}\text{-K}^t$  mapping inputs  $\varphi \in \{0, 1\}^n$  to outputs  $(x, 1^s)$  with  $|x| \geq n^{2/\delta}$  and with the superscript  $t$  greater than the running time of  $R$ . It is easy to see that, for any output  $(x, 1^s)$  of  $R(\varphi)$ ,

$$\begin{aligned} \text{K}^t(x) &\leq |\varphi| + O(\log n) \\ &\leq |x|^\delta \\ &\leq s + |x|^\delta. \end{aligned}$$

This follows from the procedure that, given  $\varphi$  hard-coded, simulates  $R(\varphi)$  and returns its output. Accordingly, a reduction of this kind cannot exist: since all of its outputs are Yes-instances, it would imply  $\varphi \in \text{SAT}$  for every formula  $\varphi$ .

When moving to the more general case of a randomized many-one reduction, one can think of  $R(\varphi)$  as a distribution over instances of  $\text{Approx}_{n^\delta}\text{-K}^t$ , and a given output  $x$  is made with probability according to  $R(\varphi)$ . Observe that in the deterministic case, it held trivially that with high probability over  $x \sim R(\varphi)$ ,

$$\text{K}^t(x) \lesssim s \iff \Pr[R(\varphi) = x] > \beta,$$

for any choice of  $\beta \in (0, 1)$ . We would like to show that something similar is true in the randomized setting. That is, there is still a correspondence between the  $\text{K}^t$  complexity of outputs and their probability under  $R(\varphi)$ . This means that  $\text{Approx}_{n^\delta}\text{-K}^t$  (and thereby SAT) will reduce to a problem of probability estimation.

There exists unconditionally a  $\text{coAM}$  protocol  $A$  that, given  $(\varphi, x, \beta)$  as input, accepts iff  $\Pr[R(\varphi) = x]$  is roughly greater than  $\beta$ , with high probability over  $x \sim R(\varphi)$  [14, 11]; see also [25, Appendix A]. Under our derandomization assumption,  $A$  can be implemented in  $\text{coNP}$ . For simplicity, assume that every output  $(x, 1^s)$  of  $R$  has the same threshold parameter  $s \in \mathbb{N}$ , so we may omit this part of the outputs. Define a parameter

$$\beta = \frac{1}{2^s \cdot \text{poly}(n)}.$$

We claim that for every  $\varphi \in \{0, 1\}^n$ ,  $A(\varphi, x, \beta)$  will work well at deciding  $\text{Approx}_{n^\delta}\text{-K}^t$  on outputs  $x$  of  $R(\varphi)$ .

On one hand, we will show that with high probability over  $x \sim R(\varphi)$ , if  $\text{K}^t(x) \leq s$ , then  $\Pr[R(\varphi) = x] > \beta$ . The idea is to use a counting argument, giving an upper bound on  $x$  such that  $\text{K}^t(x) \leq s$ , to show that  $R(\varphi)$  must be “concentrated” on these inputs. In particular, the probability over  $x \sim R(\varphi)$  that  $\text{K}^t(x) \leq s$  and  $\Pr[R(\varphi) = x] \leq \beta$  is roughly at most

$$2^s \cdot \beta = \frac{1}{\text{poly}(n)}.$$

So, with high probability over  $x \sim R(\varphi)$ , if  $x$  is a Yes-instance of  $\text{Approx}_{n^\delta}\text{-K}^t$ , then  $\Pr[R(\varphi) = x] > \beta$ , in which case  $A(\varphi, x, \beta)$  correctly outputs 1.

On the other hand, we will show that if an output  $x$  has probability greater than  $\beta$  under  $R(\varphi)$ , then  $x$  must have  $\text{K}^t$  complexity roughly upper-bounded by  $s$ . In the realm of time-unbounded Kolmogorov complexity, we could rely on the well-known Coding Theorem to prove a statement of this kind. Namely, for any samplable distribution  $D$ , it holds that

$$\text{K}(x) \leq \log(1/D(x)) + O(\log n).$$

Similarly, if  $D$  is samplable given some non-uniform input  $\varphi$ , then

$$K(x) \leq \log(1/D(x)) + |\varphi| + O(\log n).$$

Observe that our distribution  $R(\varphi)$  is samplable in polynomial time given  $\varphi$  as input. Thus, if  $x$  is samplable with probability greater than  $\beta$  under  $R(\varphi)$ , then it holds that

$$\begin{aligned} K(x) &< \log(1/\beta) + |\varphi| + O(\log n) \\ &\leq s + |\varphi| + O(\log n) \\ &\leq s + |x|^\delta. \end{aligned}$$

Of course, bounding  $K$ -complexity does not suffice for our purposes. Instead, we apply a recent work of Lu, Oliveira, and Zimand [34], which gives unconditionally a coding theorem for *probabilistic*  $K^t$  complexity, denoted  $\mathbf{p}K^t$ . Specifically, we use a version of the coding theorem for distributions samplable in polynomial time given an auxiliary non-uniform input. For some polynomial  $p_{sc}$  and time-bound  $t_0 = \text{poly}(n)$  at least the running time of  $R$ , this yields

$$\mathbf{p}K^{p_{sc}(t_0)}(x) \leq s + |\varphi| + O(\log n).$$

Roughly speaking,  $\mathbf{p}K^t$ -complexity refers to the time-bounded Kolmogorov complexity of a string in the presence of some uniform randomness. This notion is in some sense intermediate between  $K^t$  complexity and  $K$  complexity. Moreover, under the derandomization assumption  $\mathbf{E} \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ ,  $\mathbf{p}K^t$  and  $K^t$  turn out to be nearly equal: for some polynomial  $p_0$ ,  $K^{p_0(t)}(x) \leq \mathbf{p}K^t(x) + \log p_0(t)$  [16]. So, for  $t \geq p_0(p_{sc}(t_0))$ , the above implies

$$\begin{aligned} K^t(x) &\leq s + |\varphi| + O(\log n) \\ &\leq s + |x|^\delta. \end{aligned}$$

To summarize, with a sufficiently large  $t = \text{poly}(n)$  and a derandomization assumption, we obtain an auxiliary-input coding theorem for  $K^t$  complexity. This yields the required converse, namely, that high probability under  $R(\varphi)$  implies bounded  $K^t$ .<sup>4</sup>

We conclude that the  $\text{coNP}$  procedure  $A$  can be used to decide  $\text{SAT}$ . Therefore,  $\text{NP} \subseteq \text{coAM} = \text{coNP}$ .

To obtain Theorem 1 for *honest* reductions rather than polynomially length-increasing reductions, we can simply rely on the “paddability” of  $\text{SAT}$ . That is, given a  $\text{SAT}$ -instance  $\varphi \in \{0, 1\}^n$ , it is trivial to append some terms to  $\varphi$  in a way that does not affect its satisfiability but increases its length as desired. Since our assumed reduction  $R$  is honest, for some constant  $\gamma > 0$ , for any query  $x$  of  $R(\varphi)$ , it holds that  $|x| \geq |\varphi|^\gamma$ . If we let  $R'$  be the reduction that, on input  $\varphi \in \{0, 1\}^n$ , pads to obtain  $\varphi' \in \{0, 1\}^{n^{c/\gamma}}$  and then runs  $R(\varphi')$  to obtain  $x$ , we will now have  $|x| \geq |\varphi'|^\gamma = n^c$ . To summarize, if there is an honest reduction from  $\text{SAT}$  to some language  $L$ , then there is also a polynomially length-increasing reduction from  $\text{SAT}$  to  $L$ .<sup>5</sup>

For the full statement of Theorem 1, we need techniques that can handle randomized non-adaptive Turing reductions. We exploit the fact from [31] that the non-existence of a one-way function would provide an algorithm  $A$  for probability estimation as described above. In particular, for any distribution  $D \in \text{PSAMP}$ , for some poly-time computable

<sup>4</sup> We note that the use of the coding theorem for  $\mathbf{p}K^t$  is the main reason why we need to require that the runtime of our randomized  $\text{NP}$ -hardness reductions for  $\text{Approx}_{n,\delta}\text{-}K^t$  must be polynomially smaller than the parameter  $t$ .

<sup>5</sup> A similar application of padding is in [24].

function  $f$ , there is an oracle algorithm  $A$  such that  $A^I(x)$  outputs an estimate of  $\Pr[D = x]$  with high probability over  $x \sim D$ , where  $I$  is any inverter for  $f$ . Thus, in the presence of a non-adaptive reduction from SAT to  $\text{Approx}_{n^\delta}\text{-K}^t$ , we also get a non-adaptive reduction from SAT to the inversion of a one-way function. It was shown in [2, 3], with the construction of a sophisticated protocol building on techniques from [13, 11], that such a reduction would imply  $\text{SAT} \in \text{coAM}$ . However, as mentioned above, our distributions of interest  $R(\varphi)$  are not in PSAMP, but require  $\varphi$  as a non-uniform input. Luckily, a result of [9] transposes [2] to this non-uniform setting. Specifically, we have a reduction from SAT to the inversion of an *auxiliary-input* function  $f = \{f_\varphi\}_{\varphi \in \{0,1\}^*}$ , where on input  $\varphi$  to SAT, the reduction only needs to invert  $f$  on auxiliary input  $\varphi$ ; given this, [9] yields  $\text{SAT} \in \text{coAM}$ . This completes our overview of the proof of Theorem 1.

## 4.2 Proof Sketch of Theorem 2

Our proof of Theorem 2 builds on that of Theorem 1, making use of a few more ideas to obtain a reduction from NP to inversion of a standard OWF. The first idea is the fact that any inverter for an appropriate function can be used as an *errorless average-case* inverter for a desired auxiliary-input function. In particular, let  $f = \{f_\varphi\}_{\varphi \in \mathbb{N}}$  be an auxiliary-input function, and define  $g$  to be the function that randomly samples  $\varphi$  from a distribution  $D'$  and then applies  $f_\varphi$  to a uniformly random input  $z$ . It is not hard to show by an averaging argument that any inverter for  $g$  works as an inverter for  $f_\varphi$  with high probability over  $\varphi \sim D'$ . Moreover, crucially, if the inverter fails to invert some  $f_\varphi$ , then it can be made to output a special failure symbol  $\perp$  when given the auxiliary input  $\varphi$ , with high probability. This is due to the fact that successful inversion can be verified in poly-time: given a candidate pre-image  $y$  of some string  $z$  under  $f_\varphi$ , simply run  $f_\varphi(y)$  to verify; see [24, Theorem 10.3]. This, along with a reduction from SAT to inverting an auxiliary-input OWF, yields an errorless randomized heuristic for SAT over any distribution  $D' \in \text{PSAMP}$ .

The final piece of Theorem 2 is a worst-case to average-case reduction. The goal is to obtain

$$(\text{SAT}, D') \in \text{AvgBPP} \implies \text{SAT} \in \text{BPP},$$

which will complete the proof given the discussion above. To that end, we employ tools from [19] and follow-up works. A difficulty is that, from  $(\text{SAT}, D') \in \text{AvgBPP}$ , the available worst-case to average-case reductions only yield

$$\text{Gap}_{\tau, n^\delta} \text{pK}^t \in \text{BPP}.$$

The promise-problem  $\text{Gap}_{\tau, n^\delta} \text{pK}^t$  is potentially easier than  $\text{Approx}_{n^\delta}\text{-pK}^t$ , since it involves a polynomial gap  $\tau$  between time-bounds in Yes-instances and No-instances. As a result, the gap version may not be NP-hard, so its easiness would not yield  $\text{SAT} \in \text{BPP}$ . Fortunately, by a different application of the coding theorem for  $\text{pK}^t$ , we are able to show that NP-hardness of  $\text{Approx}_{n^\delta}\text{-pK}^t$  implies NP-hardness of  $\text{Gap}_{\tau, n^\delta} \text{pK}^t$ . Roughly, with high probability over the randomness of the reduction from SAT to  $\text{Approx}_{n^\delta}\text{-pK}^t$ , the  $\text{pK}^t$  complexity of queried strings will be somewhat close to their time-unbounded K complexity. Thus, granted the leeway of the  $n^\delta$  approximation term, the difference in time-bounds between  $t$  and  $\tau(t)$  does not affect the correctness of the (slightly modified) reduction when we use  $\text{Gap}_{\tau, n^\delta} \text{pK}^t$  as an oracle in lieu of  $\text{Approx}_{n^\delta}\text{-pK}^t$ .

To summarize, an outline of the proof is as follows.

1. Arguing as in Theorem 1, we get a black-box non-adaptive fixed-auxiliary input reduction from SAT to inverting an auxiliary-input function,  $f = \{f_\varphi\}_{\varphi \in \{0,1\}^*}$ .

2. Under our assumption of the non-existence of OWFs, we get, for any polynomial-time samplable distribution  $D$ , a PPT machine that inverts  $f_\varphi$  with high probability over  $\varphi \sim D$ . Combined with step (1), this yields that  $(\text{SAT}, D) \in \text{AvgBPP}$ .
3. From the worst-case to average-case reduction of [19] (and subsequent works [22] and [16]), for some distribution  $D' \in \text{PSAMP}$ , there is a BPP-black-box non-adaptive randomized polynomial-time reduction from  $\text{Gap}_{\tau, O(\log n)} \text{pK}^t$  to the average-case problem of solving SAT over  $D'$ . That is,

$$(\text{SAT}, D') \in \text{AvgBPP} \implies \text{Gap}_{\tau, O(\log n)} \text{pK}^t \in \text{BPP}$$

for a sufficiently large polynomial  $\tau$  depending on the running time of the heuristic for SAT. Combined with step (2), we get that  $\text{Gap}_{\tau, O(\log n)} \text{pK}^t \in \text{BPP}$ .

4. For a sufficiently large  $t$ , if  $\text{Approx}_{n^\delta} \text{pK}^t$  is NP-hard, then  $\text{Gap}_{\tau, O(\log n)} \text{pK}^t$  is also NP-hard. Combined with step (3), this yields  $\text{NP} \subseteq \text{BPP}$ .

### 4.3 Proof Sketch of Theorem 4

For the proof of Theorem 4 in the setting of exact  $\text{pK}^t$  and  $\text{K}^t$ , the approach discussed above does not work; recall that the approximation term  $n^\delta$  was critical at a number of points. Thus, our starting point is the following statement from a recent work of Liu and Pass [33].

Assuming  $\text{E} \not\subseteq \text{io-NSIZE}[2^{o(n)}]$ , if  $\{\text{MK}^t \text{P}\} \times \text{SAMP}[t_D(n)] \not\subseteq \text{HeurP}$  for some time bound  $t_D$  polynomially less than  $t$ , then one-way functions exist.

That is, the average-case hardness of  $\text{MK}^t \text{P}$  with respect to *any* distribution samplable within some polynomial running time smaller than  $t$  would suffice to imply one-way functions.

Our goal now is to show that if  $\text{MK}^t \text{P}$  is NP-hard, then  $\{\text{MK}^t \text{P}\} \times \text{SAMP}[t_D(n)]$  is “hard for distributional NP”: namely, if  $\text{MK}^t \text{P}$  is easy on average over every distribution  $D$  samplable in time  $t_D$ , then every distributional problem  $(L, D') \in \text{NP} \times \text{PSAMP}$  is likewise easy on average. Combining this with the statement from [33], we would get

$$\begin{aligned} \text{DistNP} \not\subseteq \text{HeurP} &\implies \{\text{MK}^t \text{P}\} \times \text{SAMP}[t_D(n)] \not\subseteq \text{HeurP} \\ &\implies \exists \text{OWF}. \end{aligned}$$

To show the distributional NP-hardness of  $\text{MK}^t \text{P}$ , we reduce from an arbitrary distributional problem  $(L, D') \in \text{DistNP}$ . Under the assumed NP-hardness of  $\text{MK}^t \text{P}$ , there is a randomized non-adaptive reduction  $R$  from  $L$  to  $\text{MK}^t \text{P}$ . With a large enough choice of the polynomial  $t$ , we can ensure that the reduction from  $L$  to  $\text{MK}^t \text{P}$  runs in time polynomially less than  $t$ . In particular, we get that the following distribution  $Q$  is samplable in time at most  $t_D$ :

Sample  $x \sim D'$ , and then output a sample from the query distribution of  $R(x)$ .

From there, it is not too hard to show that, if  $H$  is a heuristic for  $\text{MK}^t \text{P}$  working over  $Q$ , then the algorithm  $R^H$  (that simulates  $R$  and answers any oracle queries with  $H$ ) is a heuristic for  $L$  over  $D'$ . This yields the desired result.

### 4.4 Proof Sketch of Theorem 5

Finally, the proof of Theorem 5 proceeds along the lines of that of Theorem 1, but with several important changes.<sup>6</sup> The main challenge is that the Coding Theorem for  $\text{K}$  only gives us an *approximate* equality between  $\text{K}(x)$  and  $\log(1/D(x))$  for  $x$ 's sampled from a distribution

<sup>6</sup> As mentioned above, we actually give two different proofs of Theorem 5. We describe the first one here.

*D.* This was not a problem for Theorem 1 as it dealt with an *approximate* version of  $K^t$ , and we could absorb some slack of the Coding Theorem into an approximation error of  $K^t$ . But Theorem 5 is for the *exact* version of  $K$ , and we cannot apply the same strategy here. Instead, we show that this slack can be absorbed by a different argument, crucially relying on the fact that the randomized reductions  $R$  in the assumption of Theorem 5 are *many-one* and have the error probability *inverse-polynomially small* in their runtime  $t_R$ .

Namely, for  $\varphi \in \{0, 1\}^n$ , consider the distribution of queries  $(x, 1^s)$  made by the reduction  $R(\varphi)$ . We call such a query “heavy” if its probability (according to  $R(\varphi)$ ) is at least  $1/(\text{poly}(t_R(n)) \cdot 2^s)$ .

Our SAT algorithm (using a probability estimation protocol as in Theorem 1) essentially behaves as follows:

On input  $\varphi$ , sample a query  $(x, 1^s)$  according to  $R(\varphi)$ , and accept if  $(x, 1^s)$  is heavy.

For  $\varphi \notin \text{SAT}$  (which is the difficult case to analyze), heavy queries will cause our SAT algorithm to make a mistake by incorrectly accepting  $\varphi$ . We bound the error probability of our SAT algorithm by upperbounding the total probability mass of such heavy queries.

Roughly speaking, we upperbound the total probability mass of “heavy” queries  $(x, 1^s)$  by

$$\text{poly}(t_R(n)) \cdot \Pr[K(x) \leq s].$$

Note that, since  $\varphi \notin \text{SAT}$ , we have by the condition of correctness of the many-one reduction  $R$  that  $R(\varphi)$  must place a very small  $\gamma$  probability on its queries that are Yes-instances of MKP, i.e.,  $\Pr[K(x) \leq s] \leq \gamma$ . Hence, the error probability of our SAT algorithm is at most  $\text{poly}(t_R(n)) \cdot \gamma$ , which can be made sufficiently small if the error probability  $\gamma$  of the reduction  $R$  is inverse-polynomially small in the runtime  $t_R(n)$ .

## 5 NP-hardness of $(K^t \text{ vs. } K)$ and $(K^t \text{ vs. } K)^*$

In this section, we examine promise problems of the form  $(K^t \text{ vs. } K^{t'})$ , for time bounds  $t, t' \in \mathbb{N}$ , in comparison with the “partial function” versions  $(K^t \text{ vs. } K^{t'})^*$  recently shown NP-complete by Hirahara [23]. While NP-hardness of  $(K^t \text{ vs. } K)$  would imply  $\text{NP} \subseteq \text{coAM}$  via our proof techniques above, the consequence does not seem to follow in the partial setting, as we discuss further below. We then show that NP-hardness via *deterministic Turing* reductions of either  $(K^t \text{ vs. } K^{t'})$  or  $(K^t \text{ vs. } K^{t'})^*$  (with appropriate settings of  $t$  and  $t'$ ) would imply  $\text{NP} = \text{P}$ . It follows that these problems are NP-intermediate with respect to deterministic Turing reductions, provided the existence of one-way functions.

### 5.1 Randomized Reductions

We start with formal definitions of the partial version of  $K^t$  complexity and the promise problems mentioned above.

► **Definition 10** (Partial (Time-bounded) Kolmogorov Complexity). *For a time bound  $t \in \mathbb{N}$ , a string  $x \in \{0, 1, *\}^*$ , and a complexity measure  $\mu \in \{\text{p}K^t, K^t, K\}$ , the partial ( $t$ -time-bounded, probabilistic) Kolmogorov complexity of  $x$ , denoted  $(\mu)^*(x)$ , is equal to*

$$\min \{ \mu(x') \mid x' \text{ consistent with } x \},$$

where a string  $x' \in \{0, 1, *\}^*$  is said to be consistent with  $x \in \{0, 1, *\}^*$  if  $|x'| = |x|$  and, for every index  $i \in [|x|]$  such that  $x[i] \neq *$ , it holds that  $x[i] = x'[i]$ .

► **Definition 11** ( $(K^t$  vs.  $K^{t'})$ ). Let  $t, t' : \mathbb{N} \rightarrow \mathbb{N}$ . For  $\mu_1 \in \{K^t, pK^t\}$  and  $\mu_2 \in \{K^{t'}, pK^{t'}, K\}$ ,  $(\mu_1$  vs.  $\mu_2)$  is the following promise problem.

- $\Pi_Y = \{(x, 1^s) \mid \mu_1(x) \leq s\}$
- $\Pi_N = \{(x, 1^s) \mid \mu_2(x) > s\}$

$(\mu_1$  vs.  $\mu_2)^*$  is defined analogously, with the partial complexity measures  $(\mu_1)^*$  and  $(\mu_2)^*$  in place of the standard (“complete function”) ones.

By a proof analogous to that of Theorem 5, we get the following statement.

► **Lemma 12.** Let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be arbitrary and  $t_R : \mathbb{N} \rightarrow \mathbb{N}$  a polynomial. If  $(K^t$  vs.  $K)$  is NP-hard under a randomized many-one reduction running in time  $t_R(n)$  and with failure probability at most  $1/(t_R(n))^7$ , then  $\text{NP} \subseteq \text{coAM}$ .

One may contrast Lemma 12 with Hirahara’s recent proof that  $(K^t$  vs.  $K)^*$  is in fact NP-hard under a randomized many-one reduction with the same properties. This suggests that the techniques of [23] will not extend to the setting of standard  $(K^t$  vs.  $K)$  without leveraging some more powerful notion of reducibility. Viewed another way, to obtain NP-hardness of  $\text{MK}^t\text{P}$  complexity under randomized many-one reductions, one would need techniques that apply more narrowly to smaller-gap versions of the problem.

Note that the statement gives NP-hardness of  $\text{MK}^t\text{P}^*$  under a randomized reduction even when  $t \in \mathbb{N}$  is arbitrarily larger than the running time of the reduction. In the case of a randomized reduction, it is not unreasonable to make the assumption that  $t \gg t_R$ , as is done in [39] and in this work. This is because randomized reductions may easily sample strings of maximum Kolmogorov complexity, so it is easy to generate No-instances of  $\text{MK}^t\text{P}$  (or  $\text{MK}^t\text{P}^*$ ) within time  $t_R$ . Note that this would be impossible for a deterministic reduction.

► **Lemma 13** (Implicit in [23]). There exists a polynomial  $t_R : \mathbb{N} \rightarrow \mathbb{N}$  such that for any constant  $c \in \mathbb{N}$  and any sufficiently large polynomial  $t : \mathbb{N} \rightarrow \mathbb{N}$ ,  $(K^t$  vs.  $K)^*$  is NP-hard under a randomized many-one reduction running in time  $t_R(n)$  and with failure probability at most  $1/t_R(n)^c$ .

**Proof sketch.** One needs to verify that the failure probability of the reduction is at most  $1/n^c$  for an arbitrary large constant  $c \in \mathbb{N}$ . Recall that in the proof of [23] Lemma 8.3, the reduction samples random strings  $f_i \sim \{0, 1\}^{\lambda \cdot w(i)}$  for  $i \in [n]$ , where  $n \in \mathbb{N}$  is the number of variables in the input CMMSA instance,  $w : [n] \rightarrow \mathbb{N}$  is a weight function, and  $\lambda$  is some fixed polynomial in  $n$ . The reduction succeeds provided, for every  $T \subseteq [n]$ , for some constant  $c \in \mathbb{N}$ ,

$$K(f_T) \geq \lambda \cdot w(T) - c \cdot |T| \cdot \log n. \tag{1}$$

This is used in the “soundness” part of the proof to argue that the set  $B \subseteq [n]$  is not authorized. In particular, one must prove that  $w(B) < \theta$  from the fact that  $K(f_B) \leq o(\lambda \cdot w(B)) + |M|$ , where  $|M|$  is an arbitrary program of size  $\lambda\theta/2$ . To see that Eq. (1) is sufficient for this purpose, observe that for any  $c \in \mathbb{N}$ ,

$$\begin{aligned} \lambda \cdot w(B) - c \cdot |B| \cdot \log n &\leq K(f_B) \\ &\leq o(\lambda \cdot w(B)) + |M| \end{aligned}$$

implies that

$$\begin{aligned} \lambda \cdot w(B) &\leq c \cdot |B| \cdot \log n + o(\lambda \cdot w(B)) + |M| \\ &\leq o(\lambda \cdot w(B)) + |M|, \end{aligned}$$

## 51:14 Consequences of Randomized Reductions from SAT to Kolmogorov Complexity

since  $c \cdot |B| \cdot \log n \leq cn \log n = o(\lambda)$ . Thus,

$$\begin{aligned} w(B) \cdot \lambda \cdot (1 - o(1)) &\leq |M| \\ &\leq \lambda \cdot \theta/2, \end{aligned}$$

which implies that  $w(B) < \theta$ , as desired.

Now we will show that, for any  $c \in \mathbb{N}$ , Eq. (1) holds with probability at least  $1 - 1/n^{c-2}$ . First observe that by a standard counting argument, with probability  $1 - 1/n^{c-2}$ ,

$$\mathsf{K}(f_{[n]}) \geq \lambda \cdot w([n]) - (c - 2) \cdot \log n.$$

Moreover,

$$\mathsf{K}(f_{[n]}) \leq \mathsf{K}(f_T) + \lambda \cdot w([n] \setminus T) + 2 \cdot |T| \cdot \log n,$$

since one may describe  $f_{[n]}$  by describing  $f_T$ , hard-wiring  $f_{[n] \setminus T}$ , and describing the set  $T \subseteq [n]$  itself. Thus,

$$\begin{aligned} \mathsf{K}(f_T) &\geq \mathsf{K}(f_{[n]}) - \lambda \cdot w([n] \setminus T) - 2 \cdot |T| \cdot \log n \\ &\geq \lambda \cdot w([n]) - (c - 2) \cdot \log n - \lambda \cdot w([n] \setminus T) - 2 \cdot |T| \cdot \log n \\ &\geq \lambda \cdot w(T) - c \cdot |T| \cdot \log n, \end{aligned}$$

so the reduction does not fail in this case.  $\blacktriangleleft$

One may wonder why the barrier of Lemma 12 does not apply to the partial  $\mathsf{K}^t$  setting. The primary issue is that a correspondence between the compressibility of queries and their probability under the query distribution  $Q_\varphi$  appears to be missing. As a result, we cannot apply our central proof technique of reducing meta-complexity to a problem of probability estimation.

Roughly speaking, there is a difference between the Kolmogorov complexity  $\mathsf{K}(z)$  of the *description* of a query  $z := (x, 1^s)$  with  $x \in \{0, 1, *\}^*$  and the *partial complexity*  $\mathsf{K}^*(x)$  of  $x$ . By the Coding Theorem for  $\mathsf{K}$ , we still have an approximate correspondence between the logarithm of the inverse probability of (the description of the query)  $z$  output by the randomized reduction and the complexity  $\mathsf{K}(z)$ . However,  $\mathsf{K}^*(x)$  can differ significantly from  $\mathsf{K}(z)$ . For example, consider a string  $y = 0^n$ , and let  $y'$  be a uniformly random string in  $\{0, *\}^n$ . Since  $y'$  is a uniformly random string over the binary alphabet  $\{0, *\}$ , it's almost certainly true that  $\mathsf{K}(y') \geq n - O(\log n)$ . On the other hand,  $\mathsf{K}^*(y') \leq \mathsf{K}(y) \leq O(\log n)$ .

More concretely, for example, consider a reduction from SAT to the problem of approximating  $(\mathsf{K}^t)^*$  (with a fixed threshold parameter  $s \in \mathbb{N}$ ). Here, the queries  $x \in \{0, 1, *\}^*$  may contain unspecified “\*” positions. On one hand, we can use a standard coding theorem (adapted appropriately) to show that a query  $x$  having probability greater than  $\beta \approx 1/(2^s \cdot \text{poly}(n))$  under the query distribution  $Q_\varphi$  would imply that  $(\mathsf{K}^t)^*(x) \lesssim s$ .

However, the converse does not seem to hold. Previously we showed that, for strings queried in the reduction, it was unlikely for a string to be both of low complexity and low probability. This followed from a counting argument and a union bound: there are roughly at most  $2^s$  strings  $x \in \{0, 1\}^*$  with  $\mathsf{K}^t(x) \leq s$ , so the cumulative probability of strings with both this property and  $Q_\varphi(x) \leq \beta$  is at most  $1/\text{poly}(n)$ . In the case of partial  $\mathsf{K}^t$ , it is no longer true that there are “few” strings of low complexity. In particular, any one short description  $d \in \{0, 1\}^s$  can witness  $(\mathsf{K}^t)^*(x) \leq s$  for  $2^n$  distinct strings  $x \in \{0, 1, *\}^n$  (unlike standard  $\mathsf{K}^t$ , where one description only “maps” to one string). Thus, partial  $\mathsf{K}^t$  complexity is not readily connected to probability under efficiently samplable distributions, which was the key connection exploited in the previous sections.



## 5.2 Deterministic Reductions

As another point of comparison, in Lemmas 15 and 16, we show that if either of  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})$  or  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})^*$  is NP-hard with respect to deterministic adaptive Turing reductions (for a sufficiently large exponential function  $t'$ ), then one obtains the stronger consequence that  $\text{NP} = \text{P}$ . This implies that if one-way functions exist,  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})$  and  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})^*$  are both NP-intermediate with respect to deterministic Turing reductions.<sup>7</sup>

Note that Lemmas 15 and 16 hold for Turing reductions with *arbitrary* polynomial running time (i.e., less than or greater than the time-bound  $t$ ), and there is no honesty requirement. After this, we show similar results for honest reductions and superpolynomial  $t'$ .

We will use the “dream-breaker” of Bogdanov et al. [10].

► **Lemma 14** ([10]). *Suppose  $\text{NP} \neq \text{P}$ . There is an algorithm  $B$  and a universal constant  $d$  with the following properties. Let  $A$  be any poly-time algorithm that attempts to solve search-SAT and only errs by incorrectly outputting  $\perp$ .<sup>8</sup> For infinitely many  $n \in \mathbb{N}$ ,  $B(A, 1^n)$  outputs a formula  $\varphi \in \{0, 1\}^n$  and a witness  $a$  such that  $\varphi(a) = 1$  but  $A(\varphi) = \perp$ . Moreover, if  $A$  runs in time at most  $n^b$  on inputs of length  $n$ , then  $B(A, 1^n)$  runs in time at most  $(n^b)^d$ .*

► **Lemma 15.** *For every constant  $c$ , there is a constant  $c'$  with the following property. Let  $t, t' : \mathbb{N} \rightarrow \mathbb{N}$  be such that for all  $n \in \mathbb{N}$ ,  $t(n) \leq n^c$  and  $t'(n) \geq 2^{c'n}$ . Then  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})$  is NP-hard under deterministic polynomial-time Turing reductions iff  $\text{NP} = \text{P}$ .*

**Proof.** Let  $M$  be a Turing reduction from search-SAT to  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})$  running in time at most  $n^b$  on inputs of length  $n \in \mathbb{N}$ . Define a machine  $M'$  that on input  $\varphi \in \{0, 1\}^n$  simulates  $M(\varphi)$  and answers its queries as follows. If the query  $(x, 1^s)$  is such that  $s \leq 4b \log n$  and  $s \leq 2|x|$ , answer the query by brute force; otherwise simply accept the query. Note that  $M'$  runs in time at most  $n^{6bc}$ .

Let  $B$  be the refuter of Lemma 14, and let  $n \in \mathbb{N}$  and  $\varphi \in \{0, 1\}^n$  be such that  $B(M', 1^n) = (\varphi, a)$  with  $M'(\varphi) = \perp$  but  $\varphi(a) = 1$ .

Clearly, if a query  $(x, 1^s)$  is such that  $s \leq 4b \log n$  or  $2|x| < s$ ,  $M'$  answers it correctly. We now claim that for every query  $(x, 1^s)$  of  $M'(\varphi)$ , it holds that  $\mathsf{K}^{t'}(x) \leq 4b \log n$ . In particular, one may compute  $x$  from advice  $(n, i)$ , where  $x$  is the  $i^{\text{th}}$  query of  $M'(\varphi)$ , in time at most

$$(n^{6bc})^d + n^{6bc} < 2^{c' \cdot |x|},$$

assuming  $2|x| \geq s > 4b \log n$  and choosing  $c' = 4cd$ , where  $d$  is the constant from Lemma 14. For  $t' : \mathbb{N} \rightarrow \mathbb{N}$  such that  $t'(m) \geq 2^{c'm}$ , this implies

$$\mathsf{K}^{t'}(x) \leq s.$$

Thus,  $M'(\varphi)$  answers all of its queries correctly with respect to  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})$ , and

$$M'(\varphi) = M^{(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})}(\varphi) = \text{search-SAT}(\varphi),$$

a contradiction. ◀

The following statement for  $(\mathsf{K}^t \text{ vs. } \mathsf{K}^{t'})^*$  indicates that Lemma 13 makes essential use of randomness, unless  $\text{NP} = \text{P}$ .

<sup>7</sup> Since either of these problems could be used to break a cryptographic PRG, the existence of OWFs means they must not be efficiently decidable.

<sup>8</sup> Note that any poly-time algorithm may be transformed into such an algorithm by verifying any candidate satisfying assignment to the input before returning it.

► **Lemma 16.** *For every constant  $c$ , there is a constant  $c'$  with the following property. Let  $t, t' : \mathbb{N} \rightarrow \mathbb{N}$  be such that for all  $n \in \mathbb{N}$ ,  $t(n) \leq n^c$  and  $t'(n) \geq 2^{c'n}$ . Then  $(K^t \text{ vs. } K^{t'})^*$  is NP-hard under deterministic polynomial-time Turing reductions iff  $\text{NP} = \text{P}$ .*

**Proof sketch.** The proof is nearly identical to that of Lemma 15. One may still compute a string *consistent* with  $x$  from advice  $(n, i)$  by simulating the reduction, obtaining the query  $x$ , and replacing any  $*$ 's in  $x$  with 0's. Let  $\tilde{x}$  be the string  $x$  with all  $*$ 's replaced by 0's. It is easy to verify that  $(K^{t'})^*(x) \leq K^{t'}(\tilde{x}) \leq 4b \log n$ . ◀

Note that we could prove the above lemmas for  $(K^t \text{ vs. } K)$  and  $(K^t \text{ vs. } K)^*$  (that is, with time-unbounded  $K$  and  $K^*$  in  $\Pi_N$ ) without the use of a dreambreaker. If we additionally assume that the NP-hardness reductions are honest, we obtain the same results but with  $t'$  any superpolynomial function.

► **Lemma 17.** *Let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be polynomial and  $t' : \mathbb{N} \rightarrow \mathbb{N}$  superpolynomial.  $(K^t \text{ vs. } K^{t'})$  is NP-hard under honest deterministic polynomial-time Turing reductions iff  $\text{NP} = \text{P}$ .*

**Proof.** Argue as in Lemma 15. Since the reduction is honest, we have

$$|x| \geq n^\gamma$$

for some constant  $\gamma > 0$ , for any string  $x$  queried in the reduction  $M$ . Recall that any such  $x$  of  $M$  may be computed from advice  $(n, i)$  in time at most

$$\begin{aligned} (n^{6bc})^d + n^{6bc} &< n^{7bcd} \\ &\leq |x|^{7bcd/\gamma} \\ &< t'(|x|), \end{aligned}$$

as desired. ◀

► **Lemma 18.** *Let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be polynomial and  $t' : \mathbb{N} \rightarrow \mathbb{N}$  superpolynomial.  $(K^t \text{ vs. } K^{t'})^*$  is NP-hard under honest deterministic polynomial-time Turing reductions iff  $\text{NP} = \text{P}$ .*

## 6 Open Questions

We have shown various consequences of (time-bounded) Kolmogorov complexity being NP-hard under randomized notions of reducibility. Some of these consequences may be taken optimistically (Theorem 4), while others may be viewed as barriers to the kinds of NP-hardness in question (Theorems 1, 5), which include kinds of reduction that have previously been used to show NP-hardness of variants of  $K^t$  complexity (e.g., [23]).

This work leaves open a number of directions; here, we indicate a few.

1. Can we remove the requirement, in Theorems 1, 2, and 4, that the time bound  $t$  in the superscript be larger than the running time of the reduction? Recall that this requirement was due to our use of the coding theorem for  $\text{pK}^t$ .
2. Can we show consequences of randomized NP-hardness reductions to MKTP or MCSP (i.e., minimization problems for Allender's KT complexity or boolean circuit size)?
3. Can we extend Theorems 1, 2, or 4 to *adaptive* randomized Turing reductions? Note that this kind of extension is unlikely in the case of Theorem 5, given the prior work discussed in Section 3 [4, 21].
4. Can we improve Theorem 5 to hold for randomized many-one reductions with constant failure probability? In particular, can we improve the "robustness" of many-one reductions to  $K$ , as in Theorem 8, to hold for constant failure probability and exponentially small failure probability?

## References

- 1 William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. Comput. Syst. Sci.*, 42(3):327–345, 1991. doi:10.1016/0022-0000(91)90006-Q.
- 2 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006. doi:10.1145/1132516.1132614.
- 3 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: on basing one-way functions on np-hardness. 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 795–796. ACM, 2010. doi:10.1145/1806689.1806798.
- 4 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
- 5 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017. doi:10.1016/J.IC.2017.04.004.
- 6 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. *ACM Trans. Comput. Theory*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.
- 7 Eric Allender, Shuichi Hirahara, and Harsha Tirumala. Kolmogorov complexity characterizes statistical zero knowledge. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPICs*, pages 3:1–3:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ITCS.2023.3.
- 8 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Comput. Complex.*, 26(2):469–496, 2017. doi:10.1007/S00037-016-0124-0.
- 9 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 211–220. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.35.
- 10 Andrej Bogdanov, Kunal Talwar, and Andrew Wan. Hard instances for satisfiability and quasi-one-way functions. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 290–300. Tsinghua University Press, 2010. URL: <http://conference.iis.tsinghua.edu.cn/ICS2010/content/papers/23.html>.
- 11 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.
- 12 Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987. doi:10.1016/0020-0190(87)90232-8.
- 13 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.
- 14 Lance Fortnow. The complexity of perfect zero-knowledge. *Adv. Comput. Res.*, 5:327–343, 1989.
- 15 Halley Goldberg and Valentine Kabanets. Consequences of randomized reductions from SAT to time-bounded Kolmogorov complexity. *Electron. Colloquium Comput. Complex.*, TR24-120, 2024. URL: <https://ecc.weizmann.ac.il/report/2024/120/>.
- 16 Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPICs*, pages 16:1–16:60. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CCC.2022.16.

- 17 Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of SZK and NISZK. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 467–484. Springer, 1999. doi:10.1007/3-540-48405-1\_30.
- 18 Dan Gutfreund and Amnon Ta-Shma. Worst-case to average-case reductions revisited. In Moses Charikar, Klaus Jansen, Omer Reingold, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, volume 4627 of *Lecture Notes in Computer Science*, pages 569–583. Springer, 2007. doi:10.1007/978-3-540-74208-1\_41.
- 19 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00032.
- 20 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 20:1–20:47. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.20.
- 21 Shuichi Hirahara. Unexpected hardness results for kolmogorov complexity under uniform reductions. 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 1038–1051. ACM, 2020. doi:10.1145/3357713.3384251.
- 22 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. 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 292–302. ACM, 2021. doi:10.1145/3406325.3451065.
- 23 Shuichi Hirahara. Np-hardness of learning programs and partial MCSP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 968–979. IEEE, 2022. doi:10.1109/FOCS54457.2022.00095.
- 24 Shuichi Hirahara. Capturing one-way functions via np-hardness of meta-complexity. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1027–1038. ACM, 2023. doi:10.1145/3564246.3585130.
- 25 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. *Electron. Colloquium Comput. Complex.*, TR15-198, 2015. arXiv:TR15-198.
- 26 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPICs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.CCC.2016.18.
- 27 John M. Hitchcock and Aduri Pavan. On the np-completeness of the minimum circuit size problem. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPICs*, pages 236–245. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.FSTTCS.2015.236.
- 28 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and  $AC^0[p]$ . 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 34:1–34:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ITCS.2020.34.


- 29 Rahul Ilango. SAT reduces to the minimum circuit size problem with a random oracle. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 733–742. IEEE, 2023. doi:10.1109/FOCS57990.2023.00048.
- 30 Rahul Ilango, Bruno Loff, and Igor C. Oliveira. Np-hardness of circuit minimization for multi-output functions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 22:1–22:36. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.22.
- 31 Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89604.
- 32 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79. ACM, 2000. doi:10.1145/335305.335314.
- 33 Yanyi Liu and Rafael Pass. One-way functions and the hardness of (probabilistic) time-bounded kolmogorov complexity w.r.t. samplable distributions. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 645–673. Springer, 2023. doi:10.1007/978-3-031-38545-2\_21.
- 34 Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded kolmogorov complexity. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 92:1–92:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.92.
- 35 Cody D. Murray and Richard Ryan Williams. On the (non) np-hardness of computing circuit complexity. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPICs*, pages 365–380. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.CCC.2015.365.
- 36 Mikito Nanashima. On basing auxiliary-input cryptography on np-hardness via nonadaptive black-box reductions. 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 29:1–29:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ITCS.2021.29.
- 37 Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 3–17. IEEE Computer Society, 1993. doi:10.1109/ISTCS.1993.253489.
- 38 Michael E. Saks and Rahul Santhanam. Circuit lower bounds from np-hardness of MCSP under turing reductions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPICs*, pages 26:1–26:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.CCC.2020.26.
- 39 Michael E. Saks and Rahul Santhanam. On randomized reductions to the random strings. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPICs*, pages 29:1–29:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CCC.2022.29.
- 40 Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.




# Trace Reconstruction from Local Statistical Queries

Xi Chen ✉ 

Columbia University, New York, NY, USA

Anindya De ✉ 

University of Pennsylvania, Philadelphia, PA, USA

Chin Ho Lee ✉ 

North Carolina State University, Raleigh, NC, USA

Rocco A. Servedio ✉ 

Columbia University, New York, NY, USA

---

## Abstract

The goal of *trace reconstruction* is to reconstruct an unknown  $n$ -bit string  $x$  given only independent random *traces* of  $x$ , where a random trace of  $x$  is obtained by passing  $x$  through a deletion channel. A *Statistical Query* (SQ) algorithm for trace reconstruction is an algorithm which can only access statistical information about the distribution of random traces of  $x$  rather than individual traces themselves. Such an algorithm is said to be  $\ell$ -*local* if each of its statistical queries corresponds to an  $\ell$ -*junta* function over some block of  $\ell$  consecutive bits in the trace. Since several – but not all – known algorithms for trace reconstruction fall under the local statistical query paradigm, it is interesting to understand the abilities and limitations of local SQ algorithms for trace reconstruction.

In this paper we establish nearly-matching upper and lower bounds on local Statistical Query algorithms for both worst-case and average-case trace reconstruction. For the worst-case problem, we show that there is an  $\tilde{O}(n^{1/5})$ -local SQ algorithm that makes all its queries with tolerance  $\tau \geq 2^{-\tilde{O}(n^{1/5})}$ , and also that any  $\tilde{O}(n^{1/5})$ -local SQ algorithm must make some query with tolerance  $\tau \leq 2^{-\tilde{\Omega}(n^{1/5})}$ . For the average-case problem, we show that there is an  $O(\log n)$ -local SQ algorithm that makes all its queries with tolerance  $\tau \geq 1/\text{poly}(n)$ , and also that any  $O(\log n)$ -local SQ algorithm must make some query with tolerance  $\tau \leq 1/\text{poly}(n)$ .

**2012 ACM Subject Classification** Mathematics of computing → Probabilistic inference problems

**Keywords and phrases** trace reconstruction, statistical queries, algorithmic statistics

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.52

**Category** RANDOM

**Funding** *Xi Chen*: Supported by NSF grants CCF-1703925, IIS-1838154, CCF-2106429 and CCF-2107187.

*Anindya De*: Supported by NSF grants CCF-1926872, CCF-1910534 and CCF-2045128.

*Chin Ho Lee*: Supported by Madhu Sudan’s and Salil Vadhan’s Simons Investigator Awards while at Harvard University.

*Rocco A. Servedio*: Supported by NSF grants IIS-1838154, CCF-2106429, CCF-2211238 and by the Simons Collaboration on Algorithms and Geometry.

## 1 Introduction

In the *trace reconstruction* problem, the goal is to reconstruct an unknown string  $x \in \{0, 1\}^n$  given access to independent random *traces* of  $x$ , where a random trace of  $x$  is a string obtained by passing  $x$  through a deletion channel that independently deletes each bit with probability  $\delta$  and concatenates the surviving bits. Trace reconstruction has been a well-studied problem since the early 2000s [30, 29, 2], and some combinatorial variants of the problem were already considered in the 1970s [26]. Over the past decade, a wide range of algorithmic results and lower bounds have been established for many variants of the trace reconstruction problem,



© Xi Chen, Anindya De, Chin Ho Lee, and Rocco A. Servedio;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 52; pp. 52:1–52:24



Leibniz International Proceedings in Informatics

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

including worst-case [32, 19, 36, 22, 10, 11], average-case [37, 23, 24, 38], and smoothed analysis [13] versions, the low deletion rate regime [12], approximate trace reconstruction [18, 8, 9, 14], coded trace reconstruction [16, 6], variants in which different bits of the source string have different deletion probabilities [21], circular trace reconstruction [35], trace reconstruction on trees [17, 5], population recovery variants [1, 33, 34], connections to other problems such as mixture distribution learning [28], and more [20, 39].

The original, and arguably most fundamental, versions of the problem are the “worst-case” and “average-case” versions with constant deletion rate  $\delta \in (0, 1)$ . In the worst-case problem the source string  $x$  is an arbitrary (worst-case) element of  $\{0, 1\}^n$ , and in the average-case problem the source string  $x$  is selected uniformly at random from  $\{0, 1\}^n$ ; equivalently, an average-case algorithm is only required to succeed for a  $1 - o_n(1)$  fraction of all  $2^n$  possible source strings  $x \in \{0, 1\}^n$ . These two problems are the focus of our work, so in the rest of this paper we consider worst-case and average-case trace reconstruction and we always assume that the deletion rate  $\delta$  is an arbitrary (known) constant in  $(0, 1)$ .

Despite much effort, there are mildly exponential gaps between the best known upper bounds and lower bounds for both worst-case and average-case trace reconstruction. Improving on earlier  $2^{\tilde{O}(n^{1/2})}$ -trace and  $2^{\tilde{O}(n^{1/3})}$ -trace algorithms of [25, 19, 36], in [11] Chase gave an algorithm for worst-case trace reconstruction that uses  $2^{\tilde{O}(n^{1/5})}$  traces. The best known lower bound, also due to Chase [10], is  $\tilde{\Omega}(n^{3/2})$  traces (improving on earlier  $\tilde{\Omega}(n^{5/4})$  and  $\Omega(n)$  lower bounds [22, 2]). For the average-case problem, improving on earlier  $\exp(O((\log n)^{1/2}))$ -trace and  $\exp(O((\log n)^{1/3}))$ -trace algorithms [37, 23, 24], Rubinfeld [38] recently gave an  $\exp(\tilde{O}((\log n)^{1/5}))$ -trace algorithm. The best known average-case lower bound, due to Chase [10], is  $\tilde{\Omega}((\log n)^{5/2})$  traces, improving on an earlier  $\tilde{\Omega}((\log n)^{9/4})$  lower bound [22].

These substantial gaps naturally suggest the study of restricted classes of algorithms for trace reconstruction, with the hope that it may be possible to obtain sharper results. This is the starting point of our work: we propose to study the trace reconstruction problem from the vantage point of *statistical query* algorithms. As our main contribution we obtain fairly sharp upper and lower bounds on *local* statistical query algorithms for trace reconstruction, as described below.

**Statistical Query trace reconstruction algorithms.** The *Statistical Query* (SQ) model [27] was first introduced by Kearns as a means to obtain PAC learning algorithms that can tolerate random classification noise. In the decades since then, the SQ model has emerged as a major topic of study in its own right in computational learning theory and related fields such as differential privacy and optimization. An attractive feature of the SQ model is that it is powerful enough to capture state-of-the-art algorithms in a variety of different settings, yet it is also amenable to proving *unconditional* lower bounds.

SQ algorithms can only access data through noisy estimates of the expected values of user-generated query functions. In the context of trace reconstruction, an SQ oracle takes as input a bounded *query function*  $q : \{0, 1\}^n \rightarrow [-1, 1]$  and a *tolerance parameter*  $\tau \in (0, 1)$  that are provided by the reconstruction algorithm. It returns a value  $\hat{P}_q$  which satisfies  $|\hat{P}_q - P_q| \leq \tau$ , where  $P_q$  is the expected value of  $q$  on a random trace, i.e.  $P_q := \mathbf{E}_{\mathbf{y} \sim \text{Del}_\delta(x)}[q(\mathbf{y})]$ .<sup>1</sup> Thus an SQ algorithm for trace reconstruction does not receive any actual traces of  $x$ ; rather, it can only use aggregate statistical information about the overall distribution of traces.

<sup>1</sup> Since the length of each trace is at most  $n$ , we view each trace  $\mathbf{y}$  as padded with a suffix of  $n - |\mathbf{y}|$  zeros, so the argument to  $q$  is actually  $\mathbf{y}0^{n-|\mathbf{y}|}$ . This is equivalent to assuming that the  $n$ -bit source string  $x$  is padded with an infinite suffix of 0-bits.



To the best of our knowledge, the current paper is among the first works that explicitly considers the trace reconstruction problem from the perspective of statistical queries (see also [15], which we discuss in more detail below). However, in hindsight the earliest nontrivial algorithms for worst-case trace reconstruction [25, 19, 36] already made it evident that SQ algorithms – in fact, SQ algorithms which use extremely simple query functions – could be effective for trace reconstruction. The algorithms of [25, 19, 36] all work by using the traces from  $\mathbf{Del}_\delta(x)$  only to obtain high-accuracy estimates of the  $n$  values  $\mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_\delta(x)}[\mathbf{y}_i]$  for  $i \in [n]$  and then doing some subsequent computation on those estimated values; thus they correspond to SQ algorithms in which each query function is simply a Boolean dictator function, i.e. a 1-junta. On the other hand, the highly efficient average-case trace reconstruction algorithms of [37, 23, 24, 38], which use a sub-polynomial number of traces, involve various “alignment” routines which attempt to identify locations in individual received traces that correspond to specific locations in the source string. These algorithms seem to make essential use of individual traces and do not seem to be compatible with the SQ model. So given that some, but not all, known trace reconstruction algorithms correspond to SQ algorithms, it is of interest to study both the abilities and limitations of SQ algorithms for trace reconstruction.

In this work we consider a natural class of SQ algorithms, which we call  $\ell$ -local SQ algorithms. An  $\ell$ -local query function  $q : \{0, 1\}^n \rightarrow [-1, 1]$  is an  $\ell$ -junta over some  $\ell$  consecutive bits of its input string, i.e. for all  $y$ ,  $q$  satisfies  $q(y) = q'(y_i, y_{i+1}, \dots, y_{i+\ell-1})$  for some index  $i$  and some function  $q' : \{0, 1\}^\ell \rightarrow [-1, 1]$ . We say that an algorithm is an  $\ell$ -local SQ algorithm with tolerance  $\tau_0$  if all of its calls to the SQ oracle are made with  $\ell$ -local query functions and the tolerance parameter for each call is at least  $\tau_0$ .

The results of [19, 36] already show that 1-local SQ algorithms with tolerance  $\tau_0 = 2^{-\tilde{O}(n^{1/3})}$  can successfully perform worst-case trace reconstruction, and moreover [19, 36] additionally show that tolerance  $\tau_0 = 2^{-\tilde{\Omega}(n^{1/3})}$  is required for any 1-local SQ worst-case trace reconstruction algorithm. Thus, in analyzing the abilities and limitations of  $\ell$ -local algorithms for trace reconstruction for a particular value of  $\ell$ , our goal is to determine the tolerance which is necessary and sufficient for such algorithms to succeed in worst-case or average-case trace reconstruction. A simple argument which we give in Section 2.1 shows that any  $\ell$ -local SQ algorithm (which may be adaptive) using tolerance  $\tau_0$  can be converted to a nonadaptive SQ algorithm that makes at most  $n2^\ell$  queries, all of which are  $\ell$ -local “subword” queries (defined in Section 2.1) of tolerance  $\tau_0 2^{-\ell}$ . Moreover, a standard argument shows that any nonadaptive SQ algorithm which makes  $M$  statistical queries, each with tolerance at least  $\tau_0$ , can be simulated in the obvious way by a standard trace reconstruction algorithm that uses  $\text{poly}(\log M, 1/\tau_0)$  independent traces from  $\mathbf{Del}_\delta(x)$ . Thus, we will be particularly interested in identifying the value  $\ell$  of the locality parameter for which tolerance (roughly)  $2^{-\ell}$  is both necessary and sufficient for trace reconstruction. As we explain next, our main results do precisely this, for both worst-case and average-case trace reconstruction.

## 1.1 Our results

We give upper and lower bounds on local SQ algorithms for both worst-case and average-case trace reconstruction. Our upper and lower bounds match each other up to fairly small factors for both the worst-case and average-case versions of the problem.

**The worst-case problem.** Our main lower bound is the following result, which gives a lower bound on the tolerance for  $n^{1/5}$ -local SQ algorithms performing worst-case trace reconstruction:

► **Theorem 1** (Worst-case lower bound, informal version of Theorem 6). *Fix any constant deletion rate  $0 < \delta < 1$ . For  $\ell = \tilde{\Theta}(n^{1/5})$ , any  $\ell$ -local SQ algorithm for worst-case trace reconstruction must have tolerance  $\tau_0 = \exp(-\Omega(n^{1/5}))$ .*

Our algorithmic result for the worst-case problem shows that this lower bound is essentially optimal:

► **Theorem 2** (Worst-case upper bound, informal version of Theorem 15). *Fix any constant deletion rate  $0 < \delta < 1$ . There is a  $\tilde{O}(n^{1/5})$ -local SQ algorithm for the worst-case trace reconstruction problem with tolerance  $\tau_0 = \exp(-\tilde{O}(n^{1/5}))$ .*

**The average-case problem.** As mentioned earlier, the state-of-the-art average-case trace reconstruction algorithms of [37, 23, 24, 38] do not seem to be compatible with the SQ model. Recall that those algorithms use  $2^{O((\log n)^c)}$  traces, for  $c \in \{1/5, 1/3, 1/2\}$ , and thus any SQ analogue of those algorithms would have tolerance  $\approx 2^{-O((\log n)^c)}$ . We show that no  $O(\log n)$ -local (or even  $n^{0.49}$ -local) SQ algorithm for average-case trace reconstruction can succeed with such a coarse tolerance parameter:

► **Theorem 3** (Average-case lower bound, informal version of Theorem 23). *Fix any constant deletion rate  $0 < \delta < 1$ . Any  $\ell$ -local SQ algorithm for average-case trace reconstruction must have tolerance  $\tau_0 \leq \ell/\sqrt{n}$ .*

Finally, we give an average-case  $O(\log n)$ -local SQ algorithm that has inverse polynomial tolerance:

► **Theorem 4** (Average-case upper bound, informal version of Theorem 25). *Fix any constant deletion rate  $0 < \delta < 1$ . There is an  $O(\log n)$ -local SQ algorithm for average-case trace reconstruction with tolerance  $\tau_0 = 1/\text{poly}(n)$ .*

Our results can be summarized as follows: As discussed immediately before Section 1.1, we may say that an  $\ell$ -local SQ algorithm with tolerance  $\tau_0$  has overall complexity  $\text{poly}(n2^\ell, 1/\tau_0)$ . Theorems 1 and 2 together say that the optimal complexity of worst-case local SQ trace reconstruction is  $2^{\tilde{\Theta}(n^{1/5})}$ , and Theorems 3 and 4 together say that the optimal complexity of average-case local SQ trace reconstruction is  $n^{\Theta(1)}$ .

## 1.2 Discussion and techniques

**The worst-case setting.** Theorem 1 and Theorem 2 should be contrasted with recent results of Cheng et al. [15], which consider a restricted class of local SQ algorithms known as  *$\ell$ -mer based* algorithms. As defined by Mazooji and Shomorony [31], the  *$\ell$ -mer density map* is a certain vector of statistics about the frequency of length- $\ell$  subwords<sup>2</sup> of the source string  $x \in \{0, 1\}^n$ . [31] gave an algorithm which, for constant deletion rate  $0 < \delta < 1/2$ , constructs an  $\varepsilon$ -accurate (in  $\ell_\infty$  distance) estimate of the  $\ell$ -mer density map using  $\text{poly}(n, 2^\ell, 1/\varepsilon)$  traces. Cheng et al. [15] defined a trace reconstruction algorithm to be  *$\ell$ -mer based* if it only uses the  $\ell$ -mer density map of  $x$ , and observed that the algorithm of [31] (see in particular Lemma 6 of [31] and its proof) only uses local statistical information about traces, and hence is a local SQ algorithm.

<sup>2</sup> Recall that a *subword* of  $x$  is a sequence of bits that occur consecutively in  $x$ , i.e.  $x_i x_{i+1} \cdots x_{i+\ell-1}$ , whereas a *substring* of  $x$  is a subsequence of bits that need not occur consecutively, i.e.  $x_{i_1} x_{i_2} \cdots x_{i_\ell}$ .

The main result of Cheng et al. is a proof that any  $n^{1/5}$ -mer based algorithm for worst-case trace reconstruction must have tolerance  $\tau_0 = 2^{-\tilde{\Omega}(n^{1/5})}$ . Our Theorem 1 generalizes this result because it gives a lower bound for the entire class of  $n^{1/5}$ -local SQ algorithms, which includes the class of  $n^{1/5}$ -mer based algorithms by the results described above. We remark that Theorem 1 also has a shorter and simpler proof than Theorem 2 of [15].

At a high-level, we obtain Theorem 1 by a reduction to proving a 1-local SQ lower bound on  $n$ -bit source strings that are “gappy.” These are strings in which every two 1s are separated by  $\gg n^{1/5}$  zeros (see Definition 8 for the precise definition). The intuition here is that for a gappy string, any  $n^{1/5}$ -bit subword in its traces is very unlikely to contain two 1s, and so all the useful information is contained in subword queries of Hamming weight at most 1, which can then be further reduced to 1-bit queries. Then we can adapt the lower bound arguments in [19] to gappy strings to obtain our lower bound.

Turning to Theorem 2, Cheng et al. observed that the  $2^{\tilde{O}(n^{1/5})}$ -trace algorithm of [11] for worst-case trace reconstruction can be interpreted as a  $\tilde{O}(n^{1/5})$ -mer based algorithm with tolerance  $\tau_0 = 2^{-\tilde{O}(n^{1/5})}$ . By the earlier observation of Cheng et al. mentioned in the first paragraph and the [31] algorithm, which works provided that the deletion rate  $\delta$  lies in  $(0, 1/2)$ , this means that Chase’s algorithm can be expressed as a  $\tilde{O}(n^{1/5})$ -local SQ algorithm which has tolerance  $\tau_0 = 2^{-\tilde{O}(n^{1/5})}$  when  $\delta \in (0, 1/2)$ . Our Theorem 15 is based on a similar observation about Chase’s algorithm, but applied directly to the local SQ model without going through the notion of  $k$ -mer statistics. Our approach is based on techniques and arguments from [13]; using these techniques allows our argument to apply more generally to the entire range of deletion rates  $\delta \in (0, 1)$ .

**Average-case.** The average-case lower bound of Theorem 3 is proved using a fairly simple argument based on “hiding” a bit which might be either 0 or 1 in the middle of the source string. We turn to the average-case upper bound.

The average-case SQ algorithm described in Theorem 4 is obtained by adapting an algorithm for *smoothed* trace reconstruction to the SQ model. The [13] paper gives an algorithm for “smoothed” trace reconstruction, which is a generalization of the average-case trace reconstruction problem. While the algorithm of [13] only interacts with the input traces by using them to form empirical estimates of subword frequencies in traces, it is not trivially an SQ algorithm. This is because the [13] algorithm estimates these subword frequencies across a range of different deletion probabilities  $\delta, \delta + \Delta, \delta + 2\Delta, \dots$  up to  $(\delta + 1)/2$ . In the usual (non-SQ) trace reconstruction setting where traces are available, it is trivial to simulate access to  $\mathbf{Del}_{\delta'}(x)$  given access to  $\mathbf{Del}_{\delta}(x)$  for any  $\delta' > \delta$ , simply by drawing  $\mathbf{y} \sim \mathbf{Del}_{\delta}(x)$  and deleting each bit of  $\mathbf{y}$  independently with probability  $\frac{1-\delta'}{1-\delta}$ . But in the SQ setting, we only have access to statistical queries of traces drawn from  $\mathbf{Del}_{\delta}(x)$  rather than individual traces. We circumvent this issue by showing that any algorithm that makes  $\ell$ -local statistical queries with tolerance  $\tau$  to  $\mathbf{Del}_{\delta'}(x)$ , for  $\delta' > \delta$ , can be simulated by an algorithm that makes only  $\ell'$ -local statistical queries with tolerance  $\tau'$  to  $\mathbf{Del}_{\delta}(x)$ , where (roughly speaking)  $\ell' \approx \ell/(1-\delta')$  and  $\tau' = \Theta(\tau)$ . With this ingredient in hand, the algorithm of [13] is easily adapted to give Theorem 4.

### 1.3 Future work

Several natural questions suggest themselves for future work. Perhaps the foremost among these is the following: Given Theorem 2, the current state-of-the-art unrestricted algorithm for the general worst-case trace reconstruction problem is an  $\tilde{O}(n^{1/5})$ -local SQ algorithm. Might it be the case that this is in fact an optimal algorithm for trace reconstruction? We

currently seem quite far from being able to resolve this (recall that the state of the art in lower bounds for unrestricted worst-case trace reconstruction algorithms is only  $\tilde{\Omega}(n^{3/2})$  traces [11]).

A partial step towards answering the above bold question would be to establish lower bounds on general SQ algorithms for worst-case trace reconstruction, i.e. SQ algorithms that are not assumed to have bounded locality. It is difficult to imagine how queries that depend on far-separated portions of an input trace could be useful, but proving this seems quite challenging.

As a concrete first goal along these lines, a generalization of the notion of an  $\ell$ -local SQ is the notion of a *size- $s$*  SQ. A size- $s$  SQ is an SQ which asks for the expected value of some  $s$ -junta function  $q'(\mathbf{y}_{i_1}, \dots, \mathbf{y}_{i_s})$  of a random trace  $\mathbf{y}$ , but unlike an  $\ell$ -local SQ the input bits of the junta do not need to form a consecutive block of positions in  $\mathbf{y}$ . Similar to Lemma 5, a size- $s$  SQ algorithm can be assumed without loss of generality to use only query functions of the form  $\mathbf{1}[y_{i_1}, \dots, y_{i_s}] = w$  as  $(i_1, \dots, i_s)$  ranges over  $\binom{[n]}{s}$  and  $w$  ranges over  $\{0, 1\}^s$ . Even the following goal appears to be quite challenging:

Show that any SQ algorithm for the worst-case trace reconstruction problem that makes only size-2 queries must have tolerance  $\tau = 1/n^{\omega(1)}$ .

We believe that this is an interesting target problem for future work.

## 2 Preliminaries

**Notation.** Given integers  $a \leq b$  we write  $[a : b]$  to denote  $\{a, \dots, b\}$ . It will be convenient for us to index a binary string  $x \in \{0, 1\}^n$  using  $[0 : n - 1]$  as  $x = (x_0, \dots, x_{n-1})$ . We write  $\ln$  to denote natural logarithm and  $\log$  to denote logarithm to the base 2. We write  $|x|$  to denote the length of a string  $x$ .

We denote the set of non-negative integers by  $\mathbb{Z}_{\geq 0}$ . We write  $D_r(z)$  to denote the closed disk in the complex plane of radius  $r$  centered at  $z \in \mathbb{C}$ , and  $\partial D_r(z)$  to denote the circle which is the boundary of that disk.

**Subwords.** Fix a string  $x \in \{0, 1\}^n$  and an integer  $k \in [n]$ . A  $k$ -subword of  $x$  is a (contiguous) subword of  $x$  of length  $k$ , given by  $(x_a, x_{a+1}, \dots, x_{a+k-1})$  for some  $a \in [0 : n - k]$ . Given such a string  $x$  and integers  $0 \leq a < b \leq n - 1$ , we write  $x[a : b]$  to denote the subword  $(x_a, x_{a+1}, \dots, x_b)$ . For a string  $w \in \{0, 1\}^k$ , let  $\#(w, x)$  denote the number of occurrences of  $w$  as a subword of  $x$ .

**Distributions.** We use bold font letters to denote probability distributions and random variables, which should be clear from the context. We write “ $\mathbf{x} \sim \mathbf{X}$ ” to indicate that random variable  $\mathbf{x}$  is distributed according to distribution  $\mathbf{X}$ .

**Deletion channel and traces.** Throughout this paper the parameter  $\delta : 0 < \delta < 1$  denotes the *deletion probability*, and we write  $\rho$  to denote the retention probability  $\rho = 1 - \delta$ . Given a string  $x \in \{0, 1\}^n$ , we write  $\mathbf{Del}_\delta(x)$  to denote the distribution of the string that results from passing  $x$  through the  $\delta$ -deletion channel (so the distribution  $\mathbf{Del}_\delta(x)$  is supported on  $\{0, 1\}^{\leq n}$ ), and we refer to a string drawn from  $\mathbf{Del}_\delta(x)$  as a *trace* of  $x$ . Recall that a random trace  $\mathbf{y} \sim \mathbf{Del}_\delta(x)$  is obtained by independently deleting each bit of  $x$  with probability  $\delta$  and concatenating the surviving bits.<sup>3</sup>

<sup>3</sup> For simplicity in this work we assume that the deletion probability  $\delta$  is known to the reconstruction algorithm.

For  $x \in \{0, 1\}^n$  (recall that we index the bits of  $x$  as  $(x_0, \dots, x_{n-1})$ ) we view a draw of a trace  $\mathbf{y} \sim \mathbf{Del}(x)$  as corresponding to a  $\rho$ -biased random draw of a subset  $\mathbf{R} \subseteq [0 : n - 1]$ , where the elements of  $\mathbf{R}$  are the bits that are *retained* in  $x$  to obtain  $\mathbf{y}$ . So if the sorted elements of  $\mathbf{R}$  are  $\mathbf{R} = \{r_0 < r_1 < \dots < r_{m-1}\}$  for some  $m \leq n$ , then the bits of the trace  $\mathbf{y} = (\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{m-1})$  are  $\mathbf{y}_0 = x_{r_0}, \mathbf{y}_1 = x_{r_1}$ , and so on.

## 2.1 Local Statistical Query algorithms

As described earlier, an  $\ell$ -local query function  $q : \{0, 1\}^n \rightarrow [-1, 1]$  is a function

$$q(\mathbf{y}) = q'(y_i, y_{i+1}, \dots, y_{i+\ell-1})$$

for some index  $i$  and some function  $q' : \{0, 1\}^\ell \rightarrow [-1, 1]$ , i.e. a real-valued bounded  $\ell$ -junta over consecutive input variables. An algorithm is an  $\ell$ -local SQ algorithm with tolerance  $\tau_0$  if all of its calls to the SQ oracle are made with  $\ell$ -local query functions and the tolerance parameter for each call is at least  $\tau_0$ .

Let us say that an  $\ell$ -local query function is a *subword query* if it is of the form

$$q'(\mathbf{y}) = \mathbf{1}[(y_i, \dots, y_{i+\ell-1}) = w] \tag{1}$$

for some string  $w \in \{0, 1\}^\ell$ . The following simple lemma shows that without loss of generality, every  $\ell$ -local SQ algorithm makes at most  $n2^\ell$  (non-adaptive) queries, corresponding to all possible length- $\ell$  subword queries:

► **Lemma 5.** *Let  $A$  be an  $\ell$ -local SQ algorithm with tolerance  $\tau_0$  (note that  $A$  may make any number of calls to the SQ oracle and may be adaptive, i.e. the choice of later queries may depend on the responses received on earlier queries). Then there is an algorithm  $A'$  with the same behavior as  $A$  which makes  $n2^\ell$  queries (all possible length- $\ell$  subword queries), each with tolerance  $\tau_0/2^\ell$ .*

**Proof.** The algorithm  $A'$  makes all  $n2^\ell$  subword queries of the form given in Equation (1), where  $i$  ranges over  $[n]$  and  $w$  ranges over  $\{0, 1\}^\ell$ . It makes each such subword query with tolerance parameter  $\tau_0/2^\ell$ . Let  $p_{i,w} = \Pr_{\mathbf{y} \sim \mathbf{Del}(x)}[(y_i, \dots, y_{i+\ell-1}) = w]$  and let  $\hat{p}_{i,w}$  be the value received from the SQ oracle in response to the query (1), so  $|\hat{p}_{i,w} - p_{i,w}| \leq \tau_0/2^\ell$ .

Let  $(q, \tau_0)$  be any (query function, tolerance) pair that  $A$  may make in the course of its execution. We show that a  $\pm\tau_0$ -accurate estimate  $\hat{P}_q$  of  $P_q$  can be computed from the responses to the  $n2^\ell$  queries of  $A'$ . This is easily seen to imply the lemma.

Since  $A$  is  $\ell$ -local, the expected value  $P_q$  is

$$P_q = \mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_\rho(x)}[q'(\mathbf{y}_i, \dots, \mathbf{y}_{i+\ell-1})]$$

for some  $q' : \{0, 1\}^\ell \rightarrow [-1, 1]$  and some  $i \in [n]$ . Since

$$P_q = \sum_{w \in \{0, 1\}^\ell} p_{i,w} \cdot q'(w),$$

by setting  $\hat{P}_q$  to be

$$P_q = \sum_{w \in \{0, 1\}^\ell} \hat{p}_{i,w} \cdot q'(w),$$

recalling that  $|q'(w)| \leq 1$  for all  $w$ , the triangle inequality gives

$$|\widehat{P}_q - P_q| = \left| \sum_{w \in \{0,1\}^\ell} (\widehat{p}_{i,w} - p_{i,w}) \cdot q'(w) \right| \leq \max_w |q'(w)| \cdot \sum_w |\widehat{p}_{i,w} - p_{i,w}| \leq \sum_w \tau_0 / 2^\ell \leq \tau_0$$

as desired.  $\blacktriangleleft$

### 3 Worst-case lower bounds

In this section we prove the following lower bound on local SQ algorithms for the worst-case trace reconstruction problem:

► **Theorem 6 (Worst-case lower bound).** *Fix any constant deletion rate  $0 < \delta < 1$ . For a suitable absolute constant  $c_0$ , any  $c_0 n^{1/5} / (\log n)^{2/5}$ -local SQ algorithm for worst-case trace reconstruction must have tolerance  $\tau_0 < \exp(-\Omega(n^{1/5} / (\log n)^{2/5}))$ .*

**Setup.** Fix any  $0 < \delta < 1$ . For notational clarity let us write  $\ell := c_0 n^{1/5} / (\log n)^{2/5}$ . Given an  $n$ -bit source string  $x$ , an index  $i \in [0 : n - 1]$ , and an  $\ell$ -bit string  $w$ , we define the value

$$p_{x,i,w} := \Pr_{\mathbf{y} \sim \text{Del}_\delta(x)}[(\mathbf{y}_i, \dots, \mathbf{y}_{i+\ell-1}) = w], \quad (2)$$

so  $p_{x,i,w}$  is the probability that a random trace of  $x$  has  $w$  as the subword starting in position  $i$ . We refer to the vector  $(p_{x,i,w})_{i \in [0:n-1], w \in \{0,1\}^\ell}$  as the  $\ell$ -subword signature of  $x$ .

We will prove the following:

► **Lemma 7.** *For a suitable absolute constant  $c_0$ , there are distinct  $n$ -bit strings  $a \neq a' \in \{0,1\}^n$  whose  $\ell$ -subword signatures are very close to each other in  $\ell_\infty$ -distance: more precisely,*

$$\text{For all } i \in [0 : n - 1], w \in \{0,1\}^\ell, \text{ we have } |p_{a,i,w} - p_{a',i,w}| \leq \exp(-2c_0 n^{1/5} / (\log n)^{2/5}). \quad (3)$$

To see why Lemma 7 implies Theorem 6, let  $A$  be any  $\ell$ -local SQ algorithm with tolerance  $\exp(-c_0 n^{1/5} / (\log n)^{2/5})$ . By Lemma 5, there is an algorithm  $A'$  with the same behavior as  $A$  which makes only subword queries for subwords of length  $\ell$ , where each query of  $A'$  has tolerance  $\exp(-c_0 n^{1/5} / (\log n)^{2/5}) / 2^\ell > \exp(-2c_0 n^{1/5} / (\log n)^{2/5})$ . By Equation (3), a query for the value of  $p_{x,i,w}$  can be answered with the value  $q_{i,w} = \frac{p_{a,i,w} + p_{a',i,w}}{2}$  whether the source string  $x$  is  $a$  or  $a'$ . But this means that it is impossible for  $A$  to be an algorithm which successfully solves the worst-case trace reconstruction problem.

In the rest of this section, we focus on establishing Lemma 7.

We require the following simple definition:

► **Definition 8.** *Given  $t > 1$ , we say that a string  $x \in \{0,1\}^n$  is  $t$ -gappy if it is of the form*

$$x = b_0 0^{t-1} b_1 0^{t-1} \dots b_{n/t-1} 0^{t-1}$$

for some string  $b_0, b_1, \dots, b_{n/t-1} \in \{0,1\}^{n/t}$ .

Recall  $\rho = 1 - \delta$ . Fix

$$t := \frac{100 \log(n) \ell}{\rho} = \Theta(n^{1/5} (\log n)^{3/5}). \quad (4)$$

The two strings  $a, a'$  whose existence is asserted by Lemma 7 will both be  $t$ -gappy. (We note that the argument of Cheng et al. [15] also used gappy strings.)

One reason that gappy strings are useful for us because they make it very easy to handle almost all of the  $\ell$ -bit strings  $w \in \{0, 1\}^\ell$  that we need to consider in order to establish Lemma 7. To see this, observe that any string  $w$  containing at least two ones is very unlikely to be a length- $\ell$  subword of a random trace  $\mathbf{y}$ : since the source string is  $t$ -gappy, we expect consecutive ones in a random trace  $\mathbf{y}$  to be at least  $\rho t \gg \ell$  positions apart from each other. More precisely, we have the following lemma:

► **Lemma 9.** *Let  $x \in \{0, 1\}^n$  be any  $t$ -gappy string, and let  $w \in \{0, 1\}^\ell$  be any string with at least two ones. Then for any  $i \in [0 : n - 1]$  we have  $p_{x,i,w} \leq 1/n^{49\ell}$ .*

**Proof.** Fix  $0 \leq \alpha < \beta \leq \ell - 1$  to be any two positions in  $w$  such that  $w_\alpha = w_\beta = 1$ , and let  $\mathbf{y} \sim \text{Del}_\delta(x)$ . Observe that we have  $p_{x,i,w} \leq \Pr[\mathbf{y}_{i+\alpha} = \mathbf{y}_{i+\beta} = 1]$ .

Let  $\mathbf{R} = \{r_0 < r_2 < \dots < r_{m-1}\} \subseteq [0 : n - 1]$  be the  $\rho$ -biased random subset of  $[0 : n - 1]$  consisting of the indices that are retained in  $x$  to obtain  $\mathbf{y}$ . We may view the draw of  $\mathbf{R}$  as being carried out sequentially in independent stages  $0, 1, \dots$ , where in each stage  $s$  the element  $s$  is included in  $\mathbf{R}$  with probability  $\rho$ . Fix any outcome of stages  $0, 1, \dots$  up until  $r_{i+\alpha}$  has been included in  $\mathbf{R}$ . Even supposing that  $x_{r_{i+\alpha}} = 1$  (so that  $y_{i+\alpha} = 1$ ), the probability that  $x_{r_{i+\beta}} = 1$  (which equals  $\Pr[\mathbf{y}_{i+\beta} = 1]$ ) is at most (writing  $k$  for  $\beta - \alpha$ )

$$\begin{aligned} & \sum_{j \geq 1} \Pr[\text{exactly } k \text{ of the } jt \text{ indices in } r_{i+\alpha} + 1, \dots, r_{i+\alpha} + jt \text{ are retained}] & (5) \\ &= \sum_{j \geq 1} \binom{jt}{k} \rho^k \delta^{jt-k} \\ &\leq \sum_{j \geq 1} (\rho jt)^k \cdot \delta^{jt/2} = (\rho t)^k \sum_{j \geq 1} j^k \delta^{jt/2} & (\text{since } k \leq jt/2) \\ &\leq (\rho t)^\ell \sum_{j \geq 1} j^\ell (1 - \rho)^{jt/2} & (\text{using } k \leq \ell) \\ &\leq (100 \log(n)\ell)^\ell \sum_{j \geq 1} j^\ell e^{-50 \log(n)\ell j}. & (\text{by the choice of } t, \text{ and using } (1 - \rho)^{1/\rho} \leq e^{-1}) \end{aligned}$$

When  $j = 1$  the first term of the sum  $\sum_{j \geq 1} j^\ell e^{-50 \log(n)\ell j}$  is  $e^{-50 \log(n)\ell}$ . The ratio of successive terms of the sum is

$$\frac{(j+1)^\ell e^{-50 \log(n)\ell(j+1)}}{j^\ell e^{-50 \log(n)\ell j}} \leq 2^\ell e^{-50 \log(n)\ell} = (2/n^{50})^\ell \ll 1/2.$$

So the sum  $\sum_{j \geq 1} j^\ell e^{-50 \log(n)\ell j}$  is at most  $2e^{-50 \log(n)\ell} = 2/n^{50\ell}$ , and since  $100 \log(n)\ell < n$  for  $n$  sufficiently large, we get that (5)  $\leq 1/n^{49\ell}$ . It follows that  $p_{x,i,w} \leq \Pr[\mathbf{y}_{i+\alpha} = \mathbf{y}_{i+\beta} = 1] \leq 1/n^{49\ell}$  as claimed. ◀

Given Lemma 9 it remains to argue about the  $\ell + 1$  strings  $w \in \{0, 1\}^\ell$  of Hamming weight 0 or 1. We handle the weight-1 strings by reducing their analysis to the analysis of *one-bit* strings as follows: fix any  $\alpha \in [0 : \ell - 1]$  and let  $w = e_\alpha \in \{0, 1\}^\ell$  be the string with a single 1 coordinate in position  $\alpha$ . The following lemma, which we prove using Lemma 9, shows that for any gappy source string  $x$  the value of  $p_{x,i,e_\alpha}$  is very close to the expected value of a *single* location in a random trace. (A sharper bound could be obtained with a bit more work, but the bound given by Lemma 10 is sufficient for our purposes.)

► **Lemma 10.** *Let  $x \in \{0, 1\}^n$  be any  $t$ -gappy string, and let  $w = e_\alpha \in \{0, 1\}^\ell$  be the string containing a single 1 in coordinate  $\alpha$ . Then for any  $i \in [0 : n - 1]$  we have*

$$\left| \Pr_{\mathbf{y} \sim \text{Del}_\delta(x)}[\mathbf{y}_{i+\alpha} = 1] - p_{x,i,e_\alpha} \right| \leq 2^{\ell-1}/n^{49\ell}.$$

**Proof.** We have

$$\Pr_{\mathbf{y} \sim \text{Del}_\delta(x)}[\mathbf{y}_{i+\alpha} = 1] = \sum_{w \in \{0,1\}^\ell: w_\alpha=1} p_{x,i,w}, \quad \text{so}$$

$$0 \leq \Pr_{\mathbf{y} \sim \text{Del}_\delta(x)}[\mathbf{y}_{i+\alpha} = 1] - p_{x,i,e_\alpha} = \sum_{w \in \{0,1\}^\ell: w_\alpha=1, |w| \geq 2} p_{x,i,w} \leq (2^{\ell-1} - 1)/n^{49\ell}$$

where the inequality is Lemma 9.  $\blacktriangleleft$

The one remaining  $\ell$ -bit string to consider is  $w = 0^\ell$ . However, if all  $2^\ell - 1$  other strings have been handled successfully then this string is automatically handled as well:

► **Lemma 11.** Fix  $a, a' \in \{0,1\}^n$  and  $i \in [0 : n-1]$ . Suppose that for all  $w \in \{0,1\}^\ell \setminus \{0^\ell\}$  we have  $|p_{a,i,w} - p_{a',i,w}| \leq \kappa$ . Then  $|p_{a,i,0^\ell} - p_{a',i,0^\ell}| \leq (2^\ell - 1)\kappa$ .

**Proof.** This is an immediate consequence of  $\sum_{w \in \{0,1\}^\ell} p_{x,i,w} = 1$ , which holds for every  $x$  and  $i$ .  $\blacktriangleleft$

Thus, it suffices to construct two  $t$ -gappy strings  $a, a'$  whose one-bit statistics are very close:

► **Lemma 12.** For  $x \in \{0,1\}^n$  and  $i \in [0 : n-1]$  define

$$p_{x,i} := \Pr_{\mathbf{y} \sim \text{Del}_\delta(x)}[\mathbf{y}_i = 1]. \quad (6)$$

Suppose that  $a \neq a' \in \{0,1\}^n$  are two  $t$ -gappy strings such that for each  $i \in [0 : n-1]$  we have  $|p_{a,i} - p_{a',i}| \leq \exp(-\Omega(n^{1/5}/(\log n)^{2/5}))$ . Then for all  $i \in [0 : n-1]$ ,  $w \in \{0,1\}^\ell$  we have

$$|p_{a,i,w} - p_{a',i,w}| \leq 2^\ell \cdot \exp(-\Omega(n^{1/5}/(\log n)^{2/5})) + 4^\ell/n^{49\ell} \leq \exp(-2c_0 n^{1/5}/(\log n)^{2/5}).$$

**Proof.** Lemma 9 gives  $|p_{a,i,w} - p_{a',i,w}| \leq 1/n^{49\ell}$  for  $|w| \geq 2$ . Lemma 10 and the assumption on  $|p_{a,i} - p_{a',i}|$  gives  $|p_{a,i,w} - p_{a',i,w}| \leq \exp(-\Omega(n^{1/5}/(\log n)^{2/5})) + 2^\ell/n^{49\ell}$  for  $|w| = 1$ . Given these bounds, Lemma 11 gives  $|p_{a,i,0^\ell} - p_{a',i,0^\ell}| \leq 2^\ell \cdot \exp(-\Omega(n^{1/5}/(\log n)^{2/5})) + 4^\ell/n^{49\ell}$ .  $\blacktriangleleft$

### 3.1 Establishing closeness of one-bit statistics

Let us write  $p_x = (p_{x,0}, \dots, p_{x,n-1})$  to denote the  $n$ -dimensional vector in  $[0,1]^n$  whose coordinates are given by Equation (6). From the results in the previous subsection it suffice to prove the following:

► **Lemma 13.** There are two distinct  $t$ -gappy strings  $a, a' \in \{0,1\}^n$  such that for all  $i \in [0 : n-1]$  we have  $\|p_a - p_{a'}\|_\infty \leq \exp(-\Omega(n^{1/5}/(\log n)^{2/5}))$ .

This is very similar to the main lower bound statement that was established in the two works [19, 36] (independently of each other); those papers considered “one-bit statistics” which correspond precisely to our  $p_{x,i}$  quantities, and showed that there are two distinct strings  $x, x' \in \{0,1\}^n$  (not restricted to be gappy) such that  $|p_{x,i} - p_{x',i}| \leq \exp(-\Omega(n^{1/3}))$  for all  $i \in [0 : n-1]$ . In what follows we adapt their techniques to deal with  $t$ -gappy source strings.

Following [19], given a pair of source strings  $a, a' \in \{0,1\}^n$  we define the corresponding *deletion-channel polynomial* (over  $\mathbb{C}$ ) to be

$$P_{a,a'}(z) := \sum_{i=0}^{n-1} (p_{a,i} - p_{a',i}) \cdot z^i. \quad (7)$$



We have

$$\|p_a - p_{a'}\|_\infty \leq \|p_a - p_{a'}\|_1 \leq \sqrt{n} \max_{z \in \partial D_1(0)} |P_{a,a'}(z)|, \quad (8)$$

where the second inequality is by Proposition 3.5 of [19] (the proof is a simple and standard computation about complex polynomials). Thus our goal is to establish the existence of two distinct  $t$ -gappy strings  $a \neq a' \in \{0, 1\}^n$  for which  $\max_{z \in \partial D_1(0)} |P_{a,a'}(z)|$  is small. To do this, we begin by observing that since bit  $j$  of a source string ends up in location  $i$  of a trace with probability  $\binom{j}{i} \rho^{i+1} \delta^{j-i}$ , we have

$$p_{a,i} = \Pr_{\mathbf{y} \sim \text{Del}_\delta(a)}[\mathbf{y}_i = 1] = \sum_{j=0}^{n-1} \binom{j}{i} \rho^{i+1} \delta^{j-i} a_j, \text{ and hence } p_{a,i} - p_{a',i} = \sum_{j=0}^{n-1} (a_j - a'_j) \binom{j}{i} \rho^{i+1} \delta^{j-i}.$$

Hence (following [19, 36]) we get

$$\begin{aligned} P_{a,a'}(z) &= \sum_{i=0}^{n-1} \left( \sum_{j=0}^{n-1} (a_j - a'_j) \binom{j}{i} \rho^{i+1} \delta^{j-i} \right) z^i = \rho \sum_{j=0}^{n-1} (a_j - a'_j) \delta^j \sum_{i=0}^{n-1} \binom{j}{i} \left( \frac{\rho z}{\delta} \right)^i \\ &= \rho \sum_{j=0}^{n-1} (a_j - a'_j) w^j \quad (\text{taking } w = 1 - \rho + \rho z) \end{aligned} \quad (9)$$

where the last line used the binomial theorem and  $\delta = 1 - \rho$ . Now, let us write the  $t$ -gappy strings  $a, a'$  as

$$a := b_0 0^{t-1} b_1 0^{t-1} \dots b_{n/t} 0^{t-1}, \quad a' := b'_0 0^{t-1} b'_1 0^{t-1} \dots b'_{n/t} 0^{t-1} \quad (10)$$

for some  $b, b' \in \{0, 1\}^{n/t}$ . From Equation (9) we get that

$$P_{a,a'}(z) = \rho \cdot \sum_{j=0}^{n/t-1} (b_j - b'_j) w^{jt} \quad (11)$$

(the structure afforded by Equation (11) is another reason why  $t$ -gappy strings are useful for us). Since  $0 < \rho = 1 - \delta < 1$  is a constant, recalling Equation (8) our goal is to establish the existence of a string  $0^{n/t} \neq v = (v_0, \dots, v_{n/t-1}) \in \{-1, 0, 1\}^{n/t}$  such that

$$\max_{\theta \in (-\pi, \pi]} \left| \sum_{j=0}^{n/t-1} v_j \left( (1 - \rho + \rho e^{i\theta})^t \right)^j \right| \quad (12)$$

is small.

As described in Theorem 6.2 of [19], a result of Borwein and Erdélyi [3] (specifically, the first proof of Theorem 3.3 in the “special case” on p. 11 of [3]) establishes the following:

► **Theorem 14** ([3]). *There are universal constants  $c_1, c_2, c_3 > 0$  such that the following holds: For all  $0 < a \leq c_1$  there exists an integer  $2 \leq k \leq c_2/a^2$  and a nonzero vector  $u \in \{-1, 0, 1\}^{k+1}$  such that  $\max_{w \in D_{6a}(1)} \left| \sum_{j=0}^k u_j w^j \right| \leq \exp(-c_3/a)$ .*

Let  $m = n^{1/5}/(\log n)^{2/5} = 1/a$ , so  $a = 1/m = (\log n)^{2/5}/n^{-1/5}$ . Recalling Equation (4), we have that  $c_2/a^2 = c_2 m^2 \ll n/(2t)$ , so we get that there exists a vector  $0^{n/(2t)} \neq u \in \{-1, 0, 1\}^{n/(2t)}$  such that

$$\max_{w \in D_{6/m}(1)} \left| \sum_{j=0}^{n/(2t)-1} u_j w^j \right| \leq \exp(-c_3 m). \quad (13)$$

## 52:12 Trace Reconstruction from Local Statistical Queries

Routine geometry shows that if  $|\theta| \leq \frac{1}{mt}$  then  $|1 - (1 - \rho + \rho e^{i\theta})^t| \leq 6/m$ , so we get that

$$\max_{|\theta| \leq 1/(mt)} \left| \sum_{j=0}^{n/(2t)-1} u_j ((1 - \rho + \rho e^{i\theta})^t)^j \right| \leq \exp(-c_3 m). \quad (14)$$

Now we can describe our final desired string  $v \in \{-1, 0, 1\}^{n/t}$ : it is obtained by padding  $u$  with a prefix of  $n/(2t)$  many zeros. We thus have

$$\begin{aligned} (12) &= \max_{\theta \in (-\pi, \pi]} \sum_{j=0}^{n/t-1} v_j ((1 - \rho + \rho e^{i\theta})^t)^j \\ &= \max_{\theta \in (-\pi, \pi]} \left( \overbrace{(1 - \rho + \rho e^{i\theta})^{n/2}}^A \cdot \overbrace{\sum_{j=0}^{n/(2t)-1} u_j ((1 - \rho + \rho e^{i\theta})^t)^j}^B \right). \end{aligned} \quad (15)$$

Since  $|1 - \rho + \rho e^{i\theta}| \leq 1$  for all  $\theta \in (-\pi, \pi]$ , we have that  $|A|$  is always at most 1 and  $|B|$  is always at most  $n/(2t)$ . We bound Equation (15) by considering two possible ranges for  $|\theta|$ . If  $|\theta| \leq 1/(mt)$ , then since  $|A| \leq 1$ , from Equation (14) we have that  $(15) \leq 1 \cdot |B| \leq \exp(-c_3 m)$ . On the other hand, if  $|\theta| > 1/(mt)$  then since  $|B| \leq n/(2t)$  and  $\rho$  is a constant between 0 and 1, we get that  $|1 - \rho + \rho e^{i\theta}| \leq 1 - \frac{c_\rho}{(mt)^2}$ , and hence

$$(15) \leq \frac{n}{2t} \cdot |A| \leq \frac{n}{2t} \cdot \left(1 - \frac{c_\rho}{(mt)^2}\right)^{n/2} \leq \exp(-c'_\rho n/(mt)^2)$$

for two constants  $c_\rho, c'_\rho > 0$  that depend only on  $\rho$ . Since  $m = n^{1/5}/(\log n)^{2/5}$  and  $n/(mt)^2 = \Theta(n^{1/5}/(\log n)^{2/5})$ , for all  $\theta \in (-\pi, \pi]$  we have that  $(12) \leq \exp(-\Omega(n^{1/5}/(\log n)^{2/5}))$ , so the proof of Lemma 13 and hence of Theorem 6 is complete.

### 4 Worst-case upper bounds

In this section we will give a local SQ algorithm for worst-case trace reconstruction, proving Theorem 2.

► **Theorem 15** (Worst-case upper bound). *Fix any constant deletion rate  $0 < \delta < 1$ . There is a worst-case SQ trace reconstruction algorithm that makes only  $(O(n^{1/5} \log^5 n))$ -local queries with tolerance  $\tau = 2^{-O(n^{1/5} \log^5 n)}$ .*

**Overview.** As discussed in the introduction, [15] showed that the state-of-the-art worst-case trace reconstruction algorithm of Chase [11] can be interpreted as a  $\tilde{O}(n^{1/5})$ -mer based algorithm, and further observed that the work [31] implicitly showed that for deletion rate  $\delta < 1/2$ , any  $k$ -mer based algorithm only relies on local statistics of random traces. The same observation can also be inferred from the work [13]; more generally, that work implicitly showed that for *any* deletion rate  $0 < \delta < 1$  (not just  $\delta < 1/2$ ), Chase's algorithm can be interpreted as a local SQ algorithm. We obtain Theorem 15 by making this interpretation explicit, without going through the notion of  $k$ -mer statistics.

In the case of  $\delta < 1/2$ , the observation in [13, 31] is the following. Chase's algorithm is based on estimating (from below) a certain univariate polynomial  $Q_x(z_0)$  at some point  $z_0$  inside the shifted complex disc  $D := \{\frac{z-\delta}{1-\delta} : |z| \leq 1\}$ . Moreover, the degree- $\ell$  coefficient of  $Q_x$  can be estimated using  $\ell$ -local statistics. When  $\delta$  is bounded away from  $1/2$ , these

works observed that the magnitude of the degree- $\ell$  term of  $Q_x$  decays exponentially in  $\ell$ , and so the contribution from the high-degree terms is negligible and can be truncated from the evaluation.

In the case of  $\delta \geq 1/2$ , a point in  $D$  can have magnitude 1 or more, and so the high-degree terms in  $Q_x$  need not decay in magnitude. Instead of evaluating the polynomial on some point in  $D$ , [13] applies a result by Borwein, Erdélyi, and Kós [4] (see Lemma 18 below) which shows that there exists a value  $t_0$  in the real interval  $[\delta, \frac{1}{4} + \frac{3}{4}\delta]$  such that  $Q_x(t_0)$  is almost as large as  $Q_x(z_0)$ , and as a result, we can estimate the truncation of  $Q_x(t_0)$  instead.

We now proceed to a detailed proof of Theorem 15. Let  $\ell := 2n^{1/5}$ . Our  $(O(n^{1/5} \log^5 n))$ -local SQ algorithm in Theorem 15 is based on the following two lemmas. (Throughout this section, it will be more convenient for us to phrase various quantities in terms of the retention rate  $\rho = 1 - \delta$ .) For a source string  $x \in \{0, 1\}^n$  and an  $\ell$ -bit pattern  $w \in \{0, 1\}^\ell$ , let  $P_{x,w}(z, t)$  be the following bivariate polynomial:

$$P_{x,w}(z, t) := \sum_{0 \leq i_1 < \dots < i_\ell \leq n-1} \prod_{k=1}^{\ell} \mathbf{1}[x_{i_k} = w_k] z^{i_1} \cdot t^{i_\ell - i_1 - (\ell-1)}.$$

► **Lemma 16.** *For every deletion rate  $\delta \in (0, 1)$ , there is a constant  $C_\rho$  such that the following holds. For every distinct pair of source strings  $x, x' \in \{0, 1\}^n$ , there is a pattern  $w \in \{0, 1\}^\ell$ , a point  $z_0 \in \{e^{i\theta} : |\theta| \leq n^{-2/5}\} \cup [1 - \rho, 1 - \frac{3}{4}\rho]$ , and a real value  $t_0 \in [1 - \rho, 1 - \frac{3}{4}\rho]$ , such that*

$$|P_{x,w}(z_0, t_0) - P_{x',w}(z_0, t_0)| \geq \exp(-C_\rho n^{1/5} \log^5 n).$$

► **Lemma 17.** *For every deletion rate  $\delta \in (0, 1)$ , there exists an SQ algorithm that makes  $C_\rho n^{1/5} \log^5 n$ -local queries with tolerance  $\exp(-C_\rho n^{1/5} \log^5 n)$  such that for every  $w \in \{0, 1\}^\ell$ ,  $z \in \{e^{i\theta} : |\theta| \leq n^{-2/5}\} \cup [1 - \rho, 1 - \frac{3}{4}\rho]$ , and  $t \in [1 - \rho, 1 - \frac{3}{4}\rho]$  it outputs an estimate  $\widehat{P}_{x,w}(z, t)$  of  $P_{x,w}(z, t)$  that is accurate to within  $\pm 0.1 \cdot \exp(-C_\rho n^{1/5} \log^5 n)$ .*

### Our $\ell$ -local SQ algorithm (Proof of Theorem 15 assuming Lemmas 16 and 17)

Given an unknown source string  $x \in \{0, 1\}^n$ , our reconstruction algorithm enumerates every pair of distinct strings  $x_1 \neq x_2 \in \{0, 1\}^n$ . For each such pair, it considers the triple  $(w, z_0, t_0)$  for that pair whose existence is given by Lemma 16. (Hence there are at most  $2^{2n}$  many such triples  $(w, z_0, t_0)$  considered in total.) Then it uses the SQ algorithm in Lemma 17 to obtain an accurate estimate  $\widehat{P}_{x,w}(z_0, t_0)$  of  $P_{x,w}(z_0, t_0)$  for each  $w$  within an additive factor of  $\pm 0.1 \cdot \exp(-C_\rho n^{1/5} \log^5 n)$ , and outputs the  $x'$  such that  $\widehat{P}_{x,w}(z_0, t_0)$  and  $P_{x',w}(z_0, t_0)$  are  $\pm 0.5 \cdot \exp(-C_\rho n^{1/5} \log^5 n)$ -close to each other for every  $w, z_0, t_0$ . The correctness follows immediately from Lemma 16, because if  $x' \neq x$ , then by that lemma there is some  $(w, z_0, t_0)$  such that by the triangle inequality we have

$$\begin{aligned} |\widehat{P}_{x,w}(z_0, t_0) - P_{x',w}(z_0, t_0)| &\geq |P_{x,w}(z_0, t_0) - P_{x',w}(z_0, t_0)| - |\widehat{P}_{x,w}(z_0, t_0) - P_{x,w}(z_0, t_0)| \\ &\geq 0.9 \cdot \exp(-C_\rho n^{1/5} \log^5 n). \end{aligned}$$

## 4.1 Proof of Lemma 16

In this subsection we prove Lemma 16. We first recall the following result from [4].

► **Lemma 18** (Theorem 5.1 in [4]). *There are constants  $c_1, c_2 > 0$  such that for every analytic function  $f$  on the open unit disc  $\{z : |z| < 1\}$  with  $|f(z)| < \frac{1}{1-|z|}$  and every  $a \in (0, 1]$ , we have*

$$|f(0)|^{\frac{c_1}{a}} \leq \exp(c_2/a) \sup_{t \in [1-a, 1-\frac{3}{4}a]} |f(t)|.$$

Note that polynomials with coefficients bounded by 1 are clearly analytic and satisfy the condition that  $|f(z)| < \frac{1}{1-|z|}$  on the open unit disc  $\{z : |z| < 1\}$ .

We note that in the actual statement in [4, Theorem 5.1], the interval containing  $t$  is  $[1-a, 1]$ . However, a close inspection of the proof reveals that the interval can be restricted to be  $[1-a, 1 - \frac{3}{4}a]$ . Specifically, their Theorem 5.1 is based on their Corollary 5.3, which in turn is based on their Corollary 5.2, where the interval is taken to be  $[1-a, 1-a + \frac{1}{4}a]$ . A self-contained proof using essentially the same argument can also be found in [13, Theorem 9].

We further note that the difference between  $[1-a, 1]$  and  $[1 - \frac{3}{4}a]$  is crucial in showing that the contribution of the high-degree terms of the relevant polynomial (Equation (16)) is negligible. Had  $t$  been 1, then  $\frac{t-(1-\rho)}{\rho} = 1$  and there would have been no exponential decay in the high-degree terms.

Lemma 16 follows from two cases below.

**Case 1:  $x_i \neq x'_i$  for some  $0 \leq i \leq \ell - 1$**

In this case, we consider the  $\ell$ -bit pattern  $w := x[0 : \ell - 1]$ . Note that  $P_{x,w}(0, 0) - P_{x',w}(0, 0) = \mathbf{1}[x[0 : \ell - 1] = w] - \mathbf{1}[x'[0 : \ell - 1] = w] = 1$ . We now apply Lemma 18 twice. The first application is to the polynomial  $Q_1(z_1) := P_{x,w}(z_1, 0) - P_{x',w}(z_1, 0)$ , which implies that there exists some  $z_0 \in [1 - \rho, 1 - \frac{3}{4}\rho]$  such that

$$|Q_1(z_0)| \geq e^{-c_2/\rho} |Q_1(0)|^{c_1/\rho} = e^{-c_2/\rho} |P_{x,w}(0, 0) - P_{x',w}(0, 0)|^{c_1/\rho} = e^{-c_2/\rho}.$$

We now apply Lemma 18 again to the polynomial

$$Q_2(z_2) := \frac{P_{x,w}(z_0, z_2) - P_{x',w}(z_0, z_2)}{\binom{n}{\ell}}.$$

Note that all coefficients in  $Q_2$  have magnitude at most 1. This implies the existence of some  $t_0 \in [1 - \rho, 1 - \frac{3}{4}\rho]$  such that

$$\begin{aligned} |P_{x,w}(z_0, t_0) - P_{x',w}(z_0, t_0)| &= \binom{n}{\ell} |Q_2(t_0)| \geq \binom{n}{\ell} e^{-c_2/\rho} |Q_2(0)|^{c_1/\rho} = \binom{n}{\ell} e^{-c_2/\rho} \left( \frac{|Q_1(z_0)|}{\binom{n}{\ell}} \right)^{c_1/\rho} \\ &\geq \frac{e^{-\frac{c_2}{\rho} - \frac{c_1 c_2}{\rho^2}}}{\binom{n}{\ell}^{\frac{c_1}{\rho} - 1}} \geq e^{-\Omega_\rho(\ell \log n)} = e^{-\Omega_\rho(n^{1/5} \log n)}, \end{aligned}$$

where the last inequality used  $\binom{n}{\ell} \geq (n/\ell)^\ell$ , and the last equality follows from our choice of  $\ell = 2n^{1/5}$ . To conclude, there exists some  $(z_0, t_0) \in [1 - \rho, 1 - \frac{3}{4}\rho]^2$  such that  $|P_{x,w}(z_0, t_0) - P_{x',w}(z_0, t_0)| \geq \Omega_\rho \binom{n}{\ell}^{-c_1/\rho}$ .

**Case 2:  $x_i = x'_i$  for all  $0 \leq i \leq \ell - 1$**

For this case, [11, Corollary 6.1] (with the interval  $[1 - 2\rho, 1]$  replaced with  $[1 - \rho, 1 - \frac{3}{4}\rho]$ ) can be restated, using Lemma 18 in a similar fashion as Case 1, as follows:

► **Lemma 19** (Corollary 6.1 in [11], slightly rephrased and refined). *For every  $\rho > 0$ , there exists a constant  $C_\rho$  such that the following holds. Let  $\ell = 2n^{1/5}$ . For every distinct  $x, x' \in \{0, 1\}^n$  where  $x_i = x'_i$  for every  $0 \leq i < \ell - 1$ , there exists a pattern  $w \in \{0, 1\}^\ell$ , a  $z_0 = e^{i\theta}$  for some  $\theta \in [-n^{-2/5}, n^{-2/5}]$  and a  $t_0 \in [1 - \rho, 1 - \frac{3}{4}\rho]$  such that*

$$\left| \sum_{0 \leq i_1 < \dots < i_\ell \leq n-1} \left( \prod_{k=1}^{\ell} \mathbf{1}[x_{i_k} = w_k] - \prod_{k=1}^{\ell} \mathbf{1}[x'_{i_k} = w_k] \right) z_0^{i_1} \cdot t_0^{i_\ell - i_1 - (\ell-1)} \right| \geq \exp(-C_\rho n^{1/5} \log^5 n).$$

Combining the two cases proves Lemma 16.

## 4.2 Proof of Lemma 17

We now prove Lemma 17. We first state the following identity relating two multivariate polynomials, each of which is defined in terms of an arbitrary  $f : \{0, 1\}^\ell \rightarrow \mathbb{C}$ . One of these involves the evaluation of  $f$  on the  $\ell$ -bit (not necessarily consecutive) substrings of the source string  $x$ , and the other involves the expectation of  $f$  evaluated on the  $\ell$ -bit substrings of a random trace  $\mathbf{y} \sim \mathbf{Del}_\delta(x)$ . This identity has now appeared in several places such as [13, 11] (see [13, Section 5.2] for a proof).

► **Fact 20.** For every  $f : \{0, 1\}^\ell \rightarrow \mathbb{C}$ ,  $x \in \{0, 1\}^n$ ,  $\rho \in [0, 1]$ , and  $z \in \mathbb{C}^\ell$ ,

$$\begin{aligned} & \rho^\ell \sum_{0 \leq i_1 < \dots < i_\ell \leq n-1} f(x_{i_1}, \dots, x_{i_\ell}) ((1-\rho) + \rho z_1)^{i_1} \prod_{k=2}^{\ell} ((1-\rho) + \rho z_k)^{i_k - i_{k-1} - 1} \\ &= \sum_{0 \leq j_1 < \dots < j_\ell \leq n-1} \mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{1-\rho}(x)} [f(\mathbf{y}_{j_1}, \dots, \mathbf{y}_{j_\ell})] z_1^{j_1} \prod_{k=2}^{\ell} z_k^{j_k - j_{k-1} - 1}. \end{aligned}$$

Letting  $f(u_1, \dots, u_\ell)$  be the indicator function  $\mathbf{1}[u = w]$  for some pattern  $w \in \{0, 1\}^\ell$ , then performing a simple change of variable  $z_i \mapsto \frac{z_i - (1-\rho)}{\rho}$ , and then identifying the variables  $z_3, \dots, z_\ell$  with the variable  $z_2$ , we obtain the following corollary.

► **Corollary 21.** For every  $\rho \in (0, 1]$ ,  $x \in \{0, 1\}^n$ ,  $w \in \{0, 1\}^\ell$ , and  $(z_1, z_2) \in \mathbb{C}^2$ ,

$$\begin{aligned} & P_{x,w}(z_1, z_2) = \\ & \rho^{-\ell} \sum_{0 \leq j_1 < \dots < j_\ell \leq n-1} \mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{1-\rho}(x)} \left[ \prod_{k=1}^{\ell} \mathbf{1}[\mathbf{y}_{j_k} = w_k] \right] \left( \frac{z_1 - (1-\rho)}{\rho} \right)^{j_1} \left( \frac{z_2 - (1-\rho)}{\rho} \right)^{j_\ell - j_1 - (\ell-1)}. \end{aligned} \quad (16)$$

Let  $Q(z_1, z_2)$  be the bivariate polynomial on the right hand side of Equation (16). Observe that for every fixed  $z_1$ , viewing  $Q(z_1, z_2)$  as a univariate polynomial in  $z_2$ , its  $z_2$ -coefficient of degree  $d$  (a univariate polynomial in  $z_1$ ) can be estimated using  $d$ -local SQs. We will first prove that  $Q$ , as a univariate polynomial in the second variable  $z_2$ , is close to its low-degree truncation  $Q_{\leq d}$  (for a suitable choice of  $d$ ), defined by

$$Q_{\leq d}(z_1, z_2) := \rho^{-\ell} \sum_{\substack{0 \leq j_1 < \dots < j_\ell \leq n-1: \\ j_\ell - j_1 - (\ell-1) \leq d}} \mathbf{E}_{\mathbf{y}} \left[ \prod_{k=1}^{\ell} \mathbf{1}[\mathbf{y}_{j_k} = w_k] \right] \left( \frac{z_1 - (1-\rho)}{\rho} \right)^{j_1} \left( \frac{z_2 - (1-\rho)}{\rho} \right)^{j_\ell - j_1 - (\ell-1)}, \quad (17)$$

when both  $z_1, z_2$  belong to the domain in Lemma 16.

► **Claim 22.** Let  $C''_\rho$  be a constant, and  $d_0 \geq C''_\rho(\ell + n^{1/5}) + 2 \log n$ . For every  $z \in \{e^{i\theta} : |\theta| \leq n^{-2/5}\} \cup [1-\rho, 1-\frac{3}{4}\rho]$  and  $t \in [1-\rho, 1-\frac{3}{4}\rho]$ , we have  $|Q_{\leq d_0}(z, t) - Q(z, t)| \leq 4 \cdot 2^{-d_0/2}$ .

Proof. It suffices to show that for every  $d \geq d_0$ , the homogeneous degree- $d$  (in the variable  $t$ ) term of  $Q$ , that is,

$$\rho^{-\ell} \sum_{\substack{0 \leq j_1 < \dots < j_\ell \leq n-1 \\ 0 \leq j_\ell - j_1 - (\ell-1) = d}} \mathbf{E} \left[ \prod_{k=1}^{\ell} \mathbf{1}[\mathbf{y}_{j_k} = w_k] \right] \left( \frac{z - (1-\rho)}{\rho} \right)^{j_1} \left( \frac{t - (1-\rho)}{\rho} \right)^{j_\ell - j_1 - (\ell-1)}, \quad (18)$$

is bounded by  $2^{-d/2}$ , as then we have  $|Q(z, t) - Q_{\leq d_0}(z, t)| \leq \sum_{d > d_0} 2^{-d/2} = 4 \cdot 2^{-d_0/2}$ , as desired.

## 52:16 Trace Reconstruction from Local Statistical Queries

We now bound Equation (18) as follows. First, the expectation in each term of the summation can be bounded by 1. Second, writing  $z$  as  $e^{i\theta}$  for some  $|\theta| \leq n^{-2/5}$ , and using  $|\cos \theta| \geq 1 - \theta^2/2$ , we have

$$\begin{aligned} |z - (1 - \rho)|^2 &= (\cos \theta - (1 - \rho))^2 + \sin^2 \theta = 1 - 2(1 - \rho) \cos \theta + (1 - \rho)^2 \\ &= 2(1 - \rho)(1 - \cos \theta) + \rho^2 \leq (1 - \rho)\theta^2 + \rho^2. \end{aligned}$$

Using  $|\theta| \leq n^{-2/5}$  and  $j_1 \leq n$ , when  $z = e^{i\theta}$  for some  $|\theta| \leq n^{-2/5}$  we have that

$$\left| \frac{z - (1 - \rho)}{\rho} \right|^{j_1} \leq \left( 1 + (1 - \rho) \left( \frac{\theta}{\rho} \right)^2 \right)^{j_1/2} \leq e^{C'_\rho n^{1/5}} \quad (19)$$

for some constant  $C'_\rho$ . And when  $z \in [1 - \rho, 1 - \frac{3}{4}\rho]$  we have  $0 \leq \frac{z - (1 - \rho)}{\rho} \leq 1/4$  and so Equation (19) is again satisfied (with room to spare). Similarly, for  $t \in [1 - \rho, 1 - \frac{3}{4}\rho]$  we have  $0 \leq \frac{t - (1 - \rho)}{\rho} \leq 1/4$ , and so  $\left| \left( \frac{t - (1 - \rho)}{\rho} \right)^d \right| \leq 4^{-d}$ .

Finally, the number of indices  $0 \leq j_1 < \dots < j_\ell \leq n - 1$  with  $j_\ell - j_1 - (\ell - 1) = d$  is at most  $n \cdot \binom{(\ell - 2) + d}{\ell - 2} \leq n \cdot 2^{d + (\ell - 2)}$ . So the degree- $d$  term (18) can be bounded by

$$\rho^{-\ell} \cdot n \cdot 2^{d + \ell - 2} \cdot e^{C'_\rho n^{1/5}} \cdot 4^{-d} \leq n \cdot (2/\rho)^\ell \cdot e^{C'_\rho n^{1/5}} \cdot 2^{-d},$$

which is at most  $2^{-d/2}$  whenever  $d \geq C''_\rho(\ell + n^{1/5}) + 2 \log n$ , for some constant  $C''_\rho$ .  $\triangleleft$

We now describe our local SQ algorithm to approximate the low-degree polynomial  $Q_{\leq d}(z, t)$ , for any  $(z, t) \in \{e^{i\theta} : |\theta| \leq n^{-2/5}\} \cup [1 - \rho, 1 - \frac{3}{4}\rho] \times [1 - \rho, 1 - \frac{3}{4}\rho]$ . Set  $d_0 := C''_\rho n^{1/5} \log^5 n \geq C''_\rho(\ell + n^{1/5}) + 2 \log n$ . Our  $d_0$ -local algorithm makes the following  $d_0$ -local queries:

$$\mathbf{E}_{\mathbf{y} \sim \text{Del}_{1-\rho}(x)} \left[ \mathbf{1}[\mathbf{y}[j : j + d_0 - 1] = u] \right] \text{ for every } u \in \{0, 1\}^{d_0} \text{ and } j \in \{0, \dots, n - 1\}.$$

Let  $\widehat{p}_{u,j}$  be the estimate of  $\mathbf{E}[\mathbf{1}[\mathbf{y}[j : j + d_0 - 1] = u]]$  that is received as a response to the query. For every fixed tuple  $0 \leq j_1 < \dots < j_\ell \leq n - 1$  such that  $j_\ell - j_1 - (\ell - 1) \leq d_0$ , using the identity

$$\mathbf{E} \left[ \prod_{k=1}^{\ell} \mathbf{1}[\mathbf{y}_{j_k} = w_k] \right] = \sum_{u \in \{0,1\}^{d_0} : \forall k \in [\ell] : u_{j_k - j_1 + 1} = w_k} \mathbf{E} \left[ \mathbf{1}[\mathbf{y}[j_1 : j_1 + d_0 - 1] = u] \right],$$

which is a sum of  $2^{d_0 - \ell}$  terms, the algorithm computes the estimate  $\widehat{p}_{w, j_1, \dots, j_\ell}$  of  $\mathbf{E} \left[ \prod_{k=1}^{\ell} \mathbf{1}[\mathbf{y}_{j_k} = w_k] \right]$  (using the estimates  $\widehat{p}_{u, j_1}$  of  $\mathbf{E}[\mathbf{1}[\mathbf{y}[j_1 : j_1 + d_0 - 1] = u]]$ ) by

$$\widehat{p}_{w, j_1, \dots, j_\ell} := \sum_{u \in \{0,1\}^{d_0} : \forall k \in [\ell] : u_{j_k - j_1 + 1} = w_k} \widehat{p}_{u, j_1},$$

for each  $w \in \{0, 1\}^\ell$  and tuple of indices  $0 \leq j_1 < \dots < j_\ell \leq n - 1$  such that  $j_\ell - j_1 - (\ell - 1) \leq d_0$ . If the tolerance for each query is  $\tau_0$ , then the error of each estimate  $\widehat{p}_{w, j_1, \dots, j_\ell}$  is  $\pm 2^{d_0 - \ell} \cdot \tau_0$ . Finally, the algorithm computes the estimate  $\widehat{Q}_{\leq d_0}(z, t)$  of  $Q_{\leq d_0}(z, t)$  using Equation (17), as

$$\widehat{Q}_{\leq d_0}(z, t) := \rho^{-\ell} \sum_{\substack{0 \leq j_1 < \dots < j_\ell \leq n-1: \\ j_\ell - j_1 - (\ell - 1) \leq d_0}} \widehat{p}_{w, j_1, \dots, j_\ell} \left( \frac{z - (1 - \rho)}{\rho} \right)^{j_1} \left( \frac{t - (1 - \rho)}{\rho} \right)^{j_\ell - j_1 - (\ell - 1)}.$$

There are at most  $n \cdot \binom{d_0 + (\ell - 1)}{\ell - 1} \leq n \cdot 2^{d_0 + (\ell - 1)}$  such tuples. So the total error is  $n \cdot 2^{d_0 + (\ell - 1)} \cdot 2^{d_0 - \ell} \cdot \tau_0 \leq n \cdot 2^{2d_0} \cdot \tau_0$ .

By Claim 22, we have that for every  $(z, t)$  in the domain specified in Lemma 17

$$\begin{aligned} |\widehat{Q}_{\leq d_0}(z, t) - P_{x,w}(z, t)| &= |\widehat{Q}_{\leq d_0}(z, t) - Q(z, t)| \\ &\leq |\widehat{Q}_{\leq d_0}(z, t) - Q_{\leq d_0}(z, t)| + |Q_{\leq d_0}(z, t) - Q(z, t)| \\ &\leq n \cdot 2^{2d_0} \cdot \tau_0 + 4 \cdot 2^{-d_0/2} \\ &= 2^{2C''_\rho} n^{1/5} \log^5 n \tau_0 + \exp(-C''_\rho n^{1/5} \log^5 n). \end{aligned}$$

Setting the tolerance parameter  $\tau_0$  to be  $\exp(-C''_\rho n^{1/5} \log^5 n)$  proves Lemma 17.

## 5 Average-case lower bounds

► **Theorem 23** (Average-case lower bound). *Fix any constant deletion rate  $0 < \delta < 1$ . Any  $\ell$ -local SQ algorithm for average-case trace reconstruction must have tolerance  $\tau_0 \leq O(\ell/\sqrt{n})$ .*

Let  $x$  be an arbitrary fixed string in  $\{0, 1\}^n$  and let  $x'$  be the string obtained from  $x$  by flipping the bit  $x_{n/2}$  in the middle. Let  $q : \{0, 1\}^n \rightarrow [-1, 1]$  be any  $\ell$ -junta query (which is not necessarily  $\ell$ -local), i.e., there are  $0 \leq i_1 < \dots < i_\ell < n$  such that  $q(x) = q'(x_{i_1}, \dots, x_{i_\ell})$  for some  $q' : \{0, 1\}^\ell \rightarrow [-1, 1]$ . We will prove the following claim:

▷ **Claim 24.** Let  $P_q := \mathbf{E}_{\mathbf{y} \sim \text{Del}_\delta(x)}[q(\mathbf{y})]$  and  $P'_q := \mathbf{E}_{\mathbf{y} \sim \text{Del}_\delta(x')}[q(\mathbf{y})]$ . Then  $|P_q - P'_q| \leq O(\ell/\sqrt{n})$ .

*Proof.* Let  $\mathbf{R}$  be a  $\rho$ -biased random draw of a subset of  $[0 : n-1]$  with  $\mathbf{R} = \{r_0, r_1, \dots, r_{m-1}\}$  for some  $m \leq n$ . Given that the only difference between  $x$  and  $x'$  is the middle bit, we have

$$|P_q - P'_q| \leq 2 \cdot \mathbf{Pr}_{\mathbf{R}} [r_{i_j} = n/2 \text{ for some } j \in [\ell]] \leq 2 \sum_{j \in [\ell]} \mathbf{Pr}_{\mathbf{R}} [r_{i_j} = n/2].$$

Since  $\delta \in (0, 1)$  is a constant,  $\mathbf{Pr}_{\mathbf{R}}[r_i = n/2] \leq O(1/\sqrt{n})$  for any  $i$ , from which the claim follows. ◀

We now prove Theorem 23:

**Proof.** (of Theorem 23) Indeed we will show that any SQ algorithm for average-case trace reconstruction that uses  $\ell$ -junta queries with tolerance  $\tau$  must satisfy  $\tau \leq O(\ell/\sqrt{n})$ . To see this, consider any SQ algorithm for trace reconstruction that uses  $\ell$ -junta queries with tolerance  $\tau$  that is larger than the  $O(\ell/\sqrt{n})$  in Claim 24. It follows from Claim 24 that, for any string  $x \in \{0, 1\}^n$ , such an algorithm cannot distinguish between  $x$  and  $x'$ . As a result, such an algorithm fails to reconstruct  $\mathbf{x} \sim \{0, 1\}^n$  with probability at least  $1/2$ . ◀

## 6 Average-case upper bounds

► **Theorem 25** (Average-case upper bound). *Fix any constant deletion rate  $0 < \delta < 1$ . There is an SQ algorithm for average-case trace reconstruction that uses  $\ell = O(\log n)$ -local queries with tolerance  $\tau = 1/\text{poly}(n)$ .*

We will prove Theorem 25 by showing that the algorithm in [14] can be simulated with local SQ queries. To do so, we will need to recall the smoothed analysis model.

► **Definition 26.** Let  $x^{\text{worst}}$  be an unknown and arbitrary string in  $\{0,1\}^n$  and  $0 < \sigma < 1$  be a “smoothing parameter.” Let  $\mathbf{x}$  be generated by flipping every bit of  $x^{\text{worst}}$  independently with probability  $\sigma$ .

For parameters  $\eta, \tau > 0$ , a  $(T, \eta, \tau)$ -trace reconstruction algorithm in the smoothed analysis model (with smoothing parameter  $\sigma$ ) has the following guarantee: With probability at least  $1 - \eta$  (over the random generation of  $\mathbf{x}$  from  $x^{\text{worst}}$ ), it is the case that the algorithm, given access to independent traces drawn from  $\text{Del}_\delta(\mathbf{x})$ , outputs the string  $\mathbf{x}$  with probability at least  $1 - \tau$  (over the random traces drawn from  $\text{Del}_\delta(\mathbf{x})$ ). The time complexity as well as the number of traces is bounded by  $T$ .

Observe that the average case trace reconstruction setting corresponds to the smoothed analysis setting with  $\sigma = 1/2$  and  $x^{\text{worst}}$  set to the all zeros string (though any fixed choice of  $x^{\text{worst}}$  works equally well).

[14] gave a polynomial-time algorithm for trace reconstruction in the smoothed analysis setting. Taking  $\sigma = 1/2$ , the main result of [14] gives the following:

► **Theorem 27** (Theorem 1 in [14]). *There is an algorithm for trace reconstruction which for any  $\eta, \tau$  and  $\delta > 0$ , has the following guarantee: With probability  $1 - \eta$  over  $\mathbf{x}$  drawn uniformly at random from  $\{0,1\}^n$ , it is the case that the algorithm, given access to independent traces drawn from  $\text{Del}_\delta(\mathbf{x})$ , outputs the string  $\mathbf{x}$  with probability at least  $1 - \tau$  (over the random traces drawn from  $\text{Del}_\delta(\mathbf{x})$ ). Its running time and sample complexity are upper bounded by*

$$T = \left(\frac{n}{\eta}\right)^{O\left(\frac{1}{(1-\delta)} \cdot \log\left(\frac{1}{(1-\delta)}\right)\right)}.$$

We begin with a short description of the algorithm in [14] (page 27 of the Arxiv version of [14]), giving only the level of detail necessary for the current paper. We set the following parameters:

$$\begin{aligned} k &= O(\log(n/\eta)), \quad \kappa = \left(\frac{1}{n} \cdot \left(\frac{1-\delta}{2}\right)^k\right)^{O(1/(1-\delta))}, \quad \theta = (1-\delta)^2/2, \\ d &= \frac{C}{\theta} \left(\ln n + k \ln \frac{C}{\theta}\right), \quad \Delta = \frac{\kappa}{2d^2 \cdot n \cdot \binom{d+k-2}{k-2}}. \end{aligned} \quad (20)$$

Set  $L$  to be the largest integer such that  $\delta + L \cdot \delta \leq (1 + \delta)/2$ .

Given two strings  $x \in \{0,1\}^n$  and  $w \in \{0,1\}^k$ , [14] define a univariate polynomial  $\text{SW}_{x,w}(\cdot)$ . The precise formal definition of this polynomial is not important for us; rather, the following relation (Equation 6 in the Arxiv version of [14]) is sufficient for our purposes:

$$\mathbf{E}_{\mathbf{y} \sim \text{Del}_{\delta'}(x)} [\#(w, \mathbf{y})] = (1 - \delta')^k \cdot \text{SW}_{x,w}(\delta'), \quad (21)$$

where  $\#(w, \mathbf{y})$  is the number of times  $w$  appears a subword of  $\mathbf{y}$ . We remark to the reader that  $\#(w, \mathbf{y})$  is a sum of  $k$ -local query functions (we will elaborate on this shortly). The algorithm in [14] proceeds as follows:

1. Define set  $S := \{\delta, \delta + \Delta, \delta + 2\Delta, \dots, \delta + L\Delta\}$ .
2. For every  $w \in \{0,1\}^k$  and  $\delta' \in S$ , the algorithm computes  $\pm\kappa$ -accurate estimates of  $\text{SW}_{x,w}(\delta')$ , using Equation (21).
3. With these estimates of  $\text{SW}_{x,w}(\delta')$  (for  $\delta' \in S$  and  $w \in \{0,1\}^k$ ), the algorithm runs a linear program followed by a greedy algorithm to reconstruct the original string.



In particular, excluding the part of Step (2) in which the estimates of  $\text{SW}_{x,w}(\delta')$  are computed, the rest of the reconstruction algorithm is deterministic and does not use the traces. Note that the reason why the [14] algorithm does not immediately translate to a local SQ algorithm for us is the following: in our model the permissible statistical queries are with respect to  $\mathbf{y} \sim \mathbf{Del}_\delta(x)$ , whereas the [14] algorithm, as sketched above, uses estimates of probabilities corresponding to statistical queries over  $\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)$  for various values of  $\delta' > \delta$ .

Thus, to obtain a local SQ algorithm, it suffices to show that the values  $\{\text{SW}_{x,w}(\delta')\}_{\delta' \in S, w \in \{0,1\}^k}$  can be estimated using local SQ queries corresponding to  $\mathbf{Del}_\delta(x)$ . More precisely, we have the following claim whose proof is immediate from the description of the above algorithm and (21).

▷ **Claim 28.** For any  $\delta' \in S$ , to compute  $\text{SW}_{x,w}(\delta')$  to error  $\kappa$ , it is sufficient to estimate the value of  $\mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)}[\#(w, \mathbf{y})]$  up to error  $\tau'$ , where

$$\tau' := \left( \frac{1}{n} \left( \frac{1-\delta}{2} \right)^k \right)^{O(1/(1-\delta))}. \quad (22)$$

*Proof.* We need to compute  $\text{SW}_{x,w}(\delta')$  for  $\delta' \in S$  to error  $\kappa$ . By (21), it suffices to compute  $\mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)}[\#(w, \mathbf{y})]$  to error  $\kappa \cdot (1-\delta')^k$ ; noting that  $\delta' \leq (1+\delta)/2$ , the claim follows. ◀

The main technical lemma of this section is the following.

▶ **Lemma 29.** *For the parameters defined as above, the following holds: Given the values of all subword queries of length  $\ell$  with tolerance  $\tau'/2$  (with  $\tau'$  defined in (22)) corresponding to  $\mathbf{Del}_\delta(x)$ , we can compute  $\text{SW}_{x,w}(\delta')$  for all  $\delta' \in S$  and  $w \in \{0,1\}^k$  to within error  $\pm\kappa$ . Here  $\ell$  is defined to be*

$$\ell = \Theta \left( \frac{k}{1-\delta} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{\ln n}{(1-\delta)} \right).$$

Before proving this lemma, we observe that Theorem 25 follows immediately from the lemma:

**Proof.** (of Theorem 25) For any constant  $0 < \delta < 1$  and  $\eta = n^{-\Theta(1)}$ , by our choice of parameters we have  $k = O(\log n)$  (see (20)). With this choice,  $\ell = O(\log n)$  and  $\tau' = n^{-\Theta(1)}$ . By Lemma 29, using the values of all subword queries of length  $\ell$  (with tolerance  $\tau'/2$ ), we can compute  $\text{SW}_{x,w}(\delta')$  for all  $\delta' \in S$  and  $w \in \{0,1\}^k$  to within error  $\pm\kappa$ . By the guarantee of the algorithm in [14], this suffices to recover  $x$ . Thus, we get Theorem 25. ◀

## 6.1 Proof of Lemma 29

We start with the following observation.

▶ **Fact 30.** *For  $\delta' \geq \delta$ , let  $0 \leq \beta_r = (1-\delta')/(1-\delta) \leq 1$ . Then  $\mathbf{Del}_{\delta'}(x) = \mathbf{Del}_{\beta_r}(\mathbf{Del}_\delta(x))$ . In other words, we can simulate traces from the deletion channel  $\mathbf{Del}_{\delta'}(\cdot)$  by first getting a trace from  $\mathbf{Del}_\delta(\cdot)$  and then passing it through the deletion channel  $\mathbf{Del}_{\beta_r}$ .*

As stated earlier, we will assume that our original string  $x$  is padded with infinitely many 0-symbols to its right. This means that for any  $i$ , the  $i^{\text{th}}$  position of the trace is well-defined. We now consider the process of getting a trace  $\mathbf{y}$  from  $\mathbf{Del}_{\delta'}(x)$  given a trace  $\mathbf{z} \sim \mathbf{Del}_\delta(x)$ . We will do this by thinking of  $\mathbf{Del}_{\beta_r}(\cdot)$  as a “selector process”. We start with the following definition.

► **Definition 31.** For a parameter  $p \in (0, 1)$  and integers  $k > 0$  and  $\ell \geq 0$ , we define the distribution  $\text{Hypernb}(p, k, \ell)$  as follows: Define an infinite random string  $\mathbf{w} = (\mathbf{w}_0, \dots)$  in  $\{0, 1\}^*$  where each bit is independently 0 with probability  $p$  and 1 with probability  $(1 - p)$ . Let  $\mathbf{i}_s$  be the location of the  $s^{\text{th}}$  one in  $\mathbf{w}$ . Then a sample from  $\text{Hypernb}(p, k, \ell)$  is given by  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell-1})$ .

Finally, we say that an outcome from  $\text{Hypernb}(p, k, \ell)$  is  $t$ -bounded if  $|\mathbf{i}_{k+\ell-1} - \mathbf{i}_k| \leq t$ .

We note that for any fixed  $s$ , the process generating  $\mathbf{i}_s$  is *memoryless*, in the sense that for any fixed  $r$  and  $s$  (with  $r \geq s$ ), the random variable  $\mathbf{i}_r - \mathbf{i}_s$  is distributed as a negative binomial random variable.

With the above definition, we can now state the following claim:

▷ **Claim 32.** Fix  $\delta' \geq \delta$ ,  $k \geq 1$ , and  $\ell \geq 0$ . Let  $\mathbf{y} \sim \text{Del}_{\delta'}(x)$  and  $\mathbf{z} \sim \text{Del}_{\delta}(x)$ . For  $\beta_r = (1 - \delta')/(1 - \delta)$ , let  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell-1}) \sim \text{Hypernb}(\beta_r, k, \ell)$ . Then the distribution of  $(\mathbf{y}_k, \dots, \mathbf{y}_{k+\ell-1})$  is identical to the distribution of  $(\mathbf{z}_{\mathbf{i}_k}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell-1}})$ .

*Proof.* The proof is essentially obvious from Fact 30 and Definition 31. In particular, from Fact 30, given  $\mathbf{z} \sim \text{Del}_{\delta}(x)$ , to get  $\mathbf{y} \sim \text{Del}_{\delta'}(x)$ , we need to simulate the deletion channel  $\text{Del}_{\beta_r}$  on the string  $\mathbf{z}$ . By definition of the deletion channel  $\text{Del}_{\beta_r}$ , the location of the positions  $(k, \dots, k + \ell - 1)$  is given by  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell-1})$  sampled from  $\text{Hypernb}(p, k, \ell)$ . This finishes the proof. ◁

We next need a lower bound on the probability that  $\text{Hypernb}(p, k, \ell)$  is bounded. To obtain this, we first state a tail bound on negative binomial random variables:

▷ **Claim 33 ([7]).** Let  $\text{Negbin}(m, p)$  be a negative binomially distributed random variable with parameters  $m$  and  $p$ , i.e. it is the number of trials needed to get  $m$  heads from independent coin tosses with heads probability  $p$ . Then  $\mathbf{E}[\text{Negbin}(m, p)] = m/p$  and furthermore, for any  $t > 1$ ,

$$\Pr[\text{Negbin}(m, p) > tm/p] \leq \exp\left(-\frac{tm(1 - 1/t)^2}{2}\right).$$

From this we can obtain the following claim which lower bounds the probability that  $\text{Hypernb}(\beta_r, k, \ell)$  is  $s$ -bounded.

▷ **Claim 34.** For any  $k$ , an outcome  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell-1}) \sim \text{Hypernb}(\beta_r, k, \ell)$  is  $s$ -bounded with probability at least  $1 - \xi$  for  $s = t(\ell - 1)/\beta_r$ , where  $\xi = \exp(-t(\ell - 1)/8)$  for  $t \geq 2$ .

*Proof.* The gap  $\mathbf{i}_{k+\ell-1} - \mathbf{i}_k$  is a negative binomial random variable which is distributed as  $\text{Negbin}(\ell - 1, \beta_r)$ . Thus, by Claim 33, it follows that

$$\Pr\left[\mathbf{i}_{k+\ell-1} - \mathbf{i}_k > \frac{t(\ell - 1)}{\beta_r}\right] \leq \exp\left(\frac{-t(\ell - 1)(1 - 1/t)^2}{2}\right).$$

For  $t \geq 2$ , we can simplify the upper bound as

$$\Pr\left[\mathbf{i}_{k+\ell-1} - \mathbf{i}_k > \frac{t(\ell - 1)}{\beta_r}\right] \leq \exp\left(\frac{-t(\ell - 1)}{8}\right).$$

Defining  $\xi$  as  $\exp(-\frac{t(\ell-1)}{8})$ , we get the claim. ◁

We now state the following technical claim.

▷ **Claim 35.** Given the value of all  $\ell$ -local subword queries for deletion channel  $\mathbf{Del}_\delta(x)$  with tolerance  $\eta$ , we can compute the value of all  $\ell'$ -local subword queries for  $\mathbf{Del}_{\delta'}(x)$  with tolerance  $\tau'$  where

$$\eta = \tau'/2; \quad \ell = C \cdot \left( \frac{\ell'}{1-\delta'} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{\ln n}{(1-\delta')} \right), \quad \text{for a suitably large constant } C.$$

*Proof.* Fix any  $w \in \{0, 1\}^{\ell'}$  and consider the quantity

$$p'_{x,k,w} := \Pr_{\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)}[(\mathbf{y}_k, \dots, \mathbf{y}_{k+\ell'-1}) = w].$$

Then, by Claim 32, it follows that

$$p'_{x,k,w} := \Pr_{\mathbf{z} \sim \mathbf{Del}_\delta(x), (\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1}) \sim \text{Hypernb}(\beta_r, k, \ell')}[\mathbf{z}_{\mathbf{i}_1}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell'-1}} = w], \quad (23)$$

where  $\beta_r = (1-\delta')/(1-\delta)$ . Define the parameter  $t = C \cdot \left( \frac{1}{1-\delta} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{\ln n}{\ell'(1-\delta)} \right)$ , where the constant  $C$  is set so that  $\exp\left(\frac{t(\ell'-1)}{8}\right) = \frac{\tau'}{2}$ . As  $\ell = t(\ell'-1)/\beta_r$ , by Claim 34 we have

$$\Pr[\mathbf{i}_{\ell'+k-1} - \mathbf{i}_{\ell'} > \ell] \leq \exp\left(\frac{t(\ell'-1)}{8}\right) = \frac{\tau'}{2}. \quad (24)$$

Now, define  $\mathcal{E}$  as the event (over the samples  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1})$ ) that  $|\mathbf{i}_{k+\ell'-1} - \mathbf{i}_k| \leq \ell$ . We now re-express

$$\begin{aligned} p'_{x,k,w} &= \Pr_{\mathbf{z} \sim \mathbf{Del}_\delta(x), (\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1})}[\mathbf{z}_{\mathbf{i}_1}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell'-1}} = w \wedge \mathcal{E}] \\ &+ \Pr_{\mathbf{z} \sim \mathbf{Del}_\delta(x), (\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1})}[\mathbf{z}_{\mathbf{i}_1}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell'-1}} = w \wedge \bar{\mathcal{E}}]. \end{aligned}$$

From the bound (24), the second term is at most  $\tau'/2$  in magnitude and thus,

$$\left| p'_{x,k,w} - \Pr_{\mathbf{z} \sim \mathbf{Del}_\delta(x), (\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1})}[\mathbf{z}_{\mathbf{i}_1}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell'-1}} = w \wedge \mathcal{E}] \right| \leq \tau'/2.$$

Furthermore, for any particular outcome of  $(\mathbf{i}_k, \dots, \mathbf{i}_{k+\ell'-1})$  for which event  $\mathcal{E}$  happens, the quantity  $\Pr_{\mathbf{z} \sim \mathbf{Del}_\delta(x)}[\mathbf{z}_{\mathbf{i}_1}, \dots, \mathbf{z}_{\mathbf{i}_{k+\ell'-1}} = w]$  is a  $\ell$ -local subword query. Since we have the value of all  $\ell$ -local subword queries up to error  $\tau'/2$ , we can compute  $p'_{x,k,w}$  to error  $\tau'$ . ◁

**Proof.** (of Lemma 29) By Claim 32, to compute  $\text{SW}_{x,w}(\delta')$  to error  $\pm\kappa$ , it suffices to compute  $\mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)}[\#(w, \mathbf{y})]$  for every  $w \in \{0, 1\}^{\ell'}$  up to error  $\pm\tau'$  where  $\tau'$  is defined in (22). Now, by Claim 35, for any given  $\delta' \geq \delta$ , to compute  $\mathbf{E}_{\mathbf{y} \sim \mathbf{Del}_{\delta'}(x)}[\#(w, \mathbf{y})]$  to error  $\tau$ , it suffices to have the value of all  $\ell$ -local subword queries to error  $\tau'/2$  where

$$\ell = C \cdot \left( \frac{\ell'}{1-\delta'} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{\ln n}{(1-\delta')} \right).$$

Since  $\delta' \leq (1+\delta)/2$ , it follows that

$$\ell \leq C \cdot \left( \frac{2\ell'}{1-\delta} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{2 \ln n}{(1-\delta)} \right).$$

Thus, if we have the value of all  $k$ -local subword queries to error  $\tau'/2$ , where  $k$  is set to

$$k = \Theta \left( \frac{\ell'}{1-\delta} \cdot \ln \left( \frac{2}{1-\delta} \right) + \frac{\ln n}{(1-\delta)} \right),$$

we can recover  $x$ . This finishes the proof. ◀

## References

- 1 Frank Ban, Xi Chen, Adam Freilich, Rocco A. Servedio, and Sandip Sinha. Beyond trace reconstruction: Population recovery from the deletion channel. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 745–768. IEEE Computer Society, 2019.
- 2 Tuğkan Batu, Sampath Kannan, Sanjeev Khanna, and Andrew McGregor. Reconstructing strings from random traces. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 910–918, 2004.
- 3 Peter Borwein and Tamás Erdélyi. Littlewood-type polynomials on subarcs of the unit circle. *Indiana University Mathematics Journal*, 46(4):1323–1346, 1997.
- 4 Peter Borwein, Tamás Erdélyi, and Géza Kós. Littlewood-type problems on  $[0, 1]$ . *Proc. London Math. Soc. (3)*, 79(1):22–46, 1999. doi:10.1112/S0024611599011831.
- 5 Tatiana Brailovskaya and Miklós Z. Rácz. Tree trace reconstruction using substraces. *J. Appl. Probab.*, 60(2):629–641, 2023. doi:10.1017/jpr.2022.81.
- 6 Joshua Brakensiek, Ray Li, and Bruce Spang. Coded trace reconstruction in a constant number of traces. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 482–493, 2020. doi:10.1109/FOCS46700.2020.00052.
- 7 Daniel G. Brown. How I wasted too long finding a concentration inequality for sums of geometric variables. Available at <https://uwspace.uwaterloo.ca/bitstream/handle/10012/17210/negbin.pdf?sequence=1>, 2011.
- 8 Diptarka Chakraborty, Debarati Das, and Robert Krauthgamer. Approximate trace reconstruction via median string (in average-case). In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 213 of *LIPICs*, pages 11:1–11:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 9 Z. Chase and Y. Peres. Approximate trace reconstruction of random strings from a constant number of traces. Available at [arXiv:2107.06454](https://arxiv.org/abs/2107.06454), 2021.
- 10 Zachary Chase. New lower bounds for trace reconstruction. *Ann. Inst. H. Poincaré Probab. Statist.*, 57(2):627–643, 2021.
- 11 Zachary Chase. Separating words and trace reconstruction. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 21–31. ACM, 2021.
- 12 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the low deletion rate regime. In *12th Innovations in Theoretical Computer Science Conference*, volume 185, pages 20:1–20:20, 2021.
- 13 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Polynomial-time trace reconstruction in the smoothed complexity model. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 54–73, 2021.
- 14 Xi Chen, Anindya De, Chin Ho Lee, Rocco A. Servedio, and Sandip Sinha. Near-optimal average-case approximate trace reconstruction from few traces. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA 2022)*, pages 779–821, 2022.
- 15 Kuan Cheng, Elena Grigorescu, Xin Li, Madhu Sudan, and Minshen Zhu. On k-mer-based and maximum likelihood estimation algorithms for trace reconstruction. *CoRR*, abs/2308.14993, 2023. doi:10.48550/arXiv.2308.14993.
- 16 Mahdi Cheraghchi, Ryan Gabrys, Olgica Milenkovic, and João Ribeiro. Coded trace reconstruction. *IEEE Trans. Inform. Theory*, 66(10):6084–6103, 2020. doi:10.1109/TIT.2020.2996377.
- 17 Sami Davies, Miklós Z. Rácz, and Cyrus Rashtchian. Reconstructing trees from traces. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 961–978. PMLR, 2019. URL: <http://proceedings.mlr.press/v99/davies19a.html>.
- 18 Sami Davies, Miklos Z. Rácz, Cyrus Rashtchian, and Benjamin G. Schiffer. Approximate trace reconstruction: Algorithms. In *IEEE International Symposium on Information Theory*, 2021.

- 19 Anindya De, Ryan O’Donnell, and Rocco A. Servedio. Optimal mean-based algorithms for trace reconstruction. In *Proceedings of the 49th ACM Symposium on Theory of Computing (STOC)*, pages 1047–1056, 2017.
- 20 Elena Grigorescu, Madhu Sudan, and Minshen Zhu. Limitations of mean-based algorithms for trace reconstruction at small distance. In *IEEE International Symposium on Information Theory*, 2021.
- 21 Lisa Hartung, Nina Holden, and Yuval Peres. Trace reconstruction with varying deletion probabilities. In *Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2018, New Orleans, LA, USA, January 8-9, 2018.*, pages 54–61, 2018.
- 22 Nina Holden and Russell Lyons. Lower bounds for trace reconstruction. *Ann. Appl. Probab.*, 30(2):503–525, 2020. doi:10.1214/19-AAP1506.
- 23 Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 1799–1840. PMLR, 2018.
- 24 Nina Holden, Robin Pemantle, Yuval Peres, and Alex Zhai. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. *Mathematical Statistics and Learning*, 2(3/4):275–309, 2019.
- 25 Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 389–398, 2008.
- 26 V. V. Kalashnik. Reconstruction of a word from its fragments. *Computational Mathematics and Computer Science (Vychislitel’naya matematika i vychislitel’naya tekhnika)*, Kharkov, 4:56–57, 1973.
- 27 M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.
- 28 Akshay Krishnamurthy, Arya Mazumdar, Andrew McGregor, and Soumyabrata Pal. Trace reconstruction: Generalized and parameterized. In *27th Annual European Symposium on Algorithms, ESA 2019*, volume 144 of *LIPICs*, pages 68:1–68:25. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 29 Vladimir Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001.
- 30 Vladimir Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *Journal of Combinatorial Theory Series A*, 93(2):310–332, 2001.
- 31 Kayvon Mazooji and Ilan Shomorony. Substring density estimation from traces. In *IEEE International Symposium on Information Theory, ISIT 2023, Taipei, Taiwan, June 25-30, 2023*, pages 803–808. IEEE, 2023. doi:10.1109/ISIT54713.2023.10206758.
- 32 Andrew McGregor, Eric Price, and Sofya Vorotnikova. Trace reconstruction revisited. In *Proceedings of the 22nd Annual European Symposium on Algorithms*, pages 689–700, 2014.
- 33 Shyam Narayanan. Population recovery from the deletion channel: Nearly matching trace reconstruction bounds. *CoRR*, abs/2004.06828, 2020.
- 34 Shyam Narayanan. Improved algorithms for population recovery from the deletion channel. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1259–1278. SIAM, 2021. doi:10.1137/1.9781611976465.77.
- 35 Shyam Narayanan and Michael Ren. Circular Trace Reconstruction. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, pages 18:1–18:18, 2021.
- 36 Fedor Nazarov and Yuval Peres. Trace reconstruction with  $\exp(O(n^{1/3}))$  samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1042–1046, 2017.
- 37 Yuval Peres and Alex Zhai. Average-case reconstruction for the deletion channel: Subpolynomially many traces suffice. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 228–239. IEEE Computer Society, 2017.

- 38 Ittai Rubinfeld. Average-case to (shifted) worst-case reduction for the trace reconstruction problem. In *50th International Colloquium on Automata, Languages, and Programming*, volume 261 of *LIPICs. Leibniz Int. Proc. Inform.*, pages Art. No. 102, 20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/lipics.icalp.2023.102.
- 39 Jin Sima and Jehoshua Bruck. Trace reconstruction with bounded edit distance. In *IEEE International Symposium on Information Theory*, 2021. Manuscript, available at [arXiv:2102.05372](https://arxiv.org/abs/2102.05372).

# When Do Low-Rate Concatenated Codes Approach The Gilbert–Varshamov Bound?

Dean Doron   

Ben-Gurion University of the Negev, Beersheba, Israel

Jonathan Mosheiff   

Ben-Gurion University of the Negev, Beersheba, Israel

Mary Wootters  

Stanford University, CA, USA

---

## Abstract

The *Gilbert–Varshamov* (GV) bound is a classical existential result in coding theory. It implies that a random linear binary code of rate  $\varepsilon^2$  has relative distance at least  $\frac{1}{2} - O(\varepsilon)$  with high probability. However, it is a major challenge to construct *explicit* codes with similar parameters.

One hope to derandomize the Gilbert–Varshamov construction is with code concatenation: We begin with a (hopefully explicit) outer code  $\mathcal{C}_{\text{out}}$  over a large alphabet, and concatenate that with a small binary random linear code  $\mathcal{C}_{\text{in}}$ . It is known that when we use *independent* small codes for each coordinate, then the result lies on the GV bound with high probability, but this still uses a lot of randomness. In this paper, we consider the question of whether code concatenation with a *single* random linear inner code  $\mathcal{C}_{\text{in}}$  can lie on the GV bound; and if so what conditions on  $\mathcal{C}_{\text{out}}$  are sufficient for this.

We show that first, there *do* exist linear outer codes  $\mathcal{C}_{\text{out}}$  that are “good” for concatenation in this sense (in fact, *most* linear codes codes are good). We also provide two sufficient conditions for  $\mathcal{C}_{\text{out}}$ , so that if  $\mathcal{C}_{\text{out}}$  satisfies these,  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  will likely lie on the GV bound. We hope that these conditions may inspire future work towards constructing explicit codes  $\mathcal{C}_{\text{out}}$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Error-correcting codes

**Keywords and phrases** Error-correcting codes, Concatenated codes, Derandomization, Gilbert–Varshamov bound

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.53

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2405.08584>

**Funding** *Dean Doron*: Supported in part by NSF-BSF grant #2022644.

*Jonathan Mosheiff*: Supported by an Alon Fellowship.

*Mary Wootters*: Partially supported by NSF grants CCF-2231157 and CCF-2133154.

**Acknowledgements** We thank Amnon Ta-Shma for helpful and interesting discussions, and collaboration at the beginning of this work. We thank Arya Mazumdar for pointing out [4] and for helping us understand its implications. This work was done partly while the authors were visiting the Simons Institute for the Theory of Computing.

## 1 Introduction

An *error correcting code* (or just a *code*) is a subset  $\mathcal{C} \subseteq \Sigma^n$ , for some alphabet  $\Sigma$ . We think of a code  $\mathcal{C}$  being used to encode messages in  $\Sigma^k$  for  $k = \log_{|\Sigma|} |\mathcal{C}|$ . That is, for any  $m \in \Sigma^k$ , we can identify  $m$  with a codeword  $\mathcal{C}(m) \in \mathcal{C}$ .<sup>1</sup> The idea is that encoding  $m$  into the

---

<sup>1</sup> Here and throughout the paper, we will abuse notation and use  $\mathcal{C}$  both as the code itself (a subset of  $\Sigma^n$ ) and also as an encoding map  $\mathcal{C}: \Sigma^k \rightarrow \Sigma^n$ .



codeword  $\mathcal{C}(m)$  will introduce redundancy that can later be used to correct errors. In this work we focus on *linear* codes  $\mathcal{C}$ , which are codes where  $\Sigma = \mathbb{F}$  is a finite field and  $\mathcal{C} \subseteq \mathbb{F}^n$  is a linear subspace of  $\mathbb{F}^n$ .

Two important properties of error correcting codes are the *rate*  $R$  and the *relative distance*  $\delta$ . For a code  $\mathcal{C} \subseteq \Sigma^n$ , the rate is defined as  $R = \frac{\log_{|\Sigma|} |\mathcal{C}|}{n} = \frac{k}{n}$ , and it quantifies how large the code is. The rate is between 0 and 1, and typically we want it to be as close to 1 as possible; this means that the encoding map does not introduce much redundancy. The (relative) distance of  $\mathcal{C} \subseteq \Sigma^n$  is defined as  $\delta = \frac{1}{n} \min_{c \neq c' \in \mathcal{C}} \Delta(c, c')$ , where  $\Delta(\cdot, \cdot)$  is Hamming distance. Again, the relative distance is between 0 and 1, and again we typically want it to be as close to 1 as possible; this means that the code can correct many worst-case errors.

These two quantities – rate and distance – are in tension. The larger the rate is, the smaller the distance must be. For binary codes (that is, codes where  $\Sigma = \mathbb{F}_2$ ), it is a major open question to pin down the best trade-off possible between rate and distance. However, we know that good trade-offs are possible: The best known possibility result in general is the *Gilbert–Varshamov* (GV) bound (Theorem 2.1 in the full version).

In this paper we focus on *low rate* codes. In this parameter regime, the GV bound implies that there *exist* binary linear codes with relative distance  $\frac{1-\varepsilon}{2}$  and rate  $\Omega(\varepsilon^2)$ , for small  $\varepsilon > 0$ . In fact, Varshamov’s proof shows that a random binary linear code achieves this with high probability.

Constructing such codes explicitly, hopefully accompanied by an efficient decoding algorithm, has been subject to extensive and fruitful research in the past decades (e.g., [24, 2, 3, 6, 11, 28, 7]), with several exciting breakthroughs in recent years. These breakthroughs include explicit constructions of codes with distance  $\delta = \frac{1-\varepsilon}{2}$  and rate  $R = \Omega(\varepsilon^{2+o(1)})$ , even with efficient algorithms (see Section 1.1). However, there are still open questions. For example, we do not know how to attain  $\delta = \frac{1-\varepsilon}{2}$  and  $R = \Omega(\varepsilon^2)$  (without any  $o(1)$  term) explicitly, and we do not have explicit constructions approaching the GV bound with rates bounded away from zero. Motivated by these questions, we consider *concatenated codes*, possibly with some randomness, which we discuss next.

### Concatenated Codes, and Our Question

A natural candidate for explicit (for low randomness) codes on the GV bound are *concatenated linear codes*. These codes are built out of two ingredients: a (hopefully explicit) linear outer code  $\mathcal{C}_{\text{out}} \subseteq \mathbb{F}_q^n$  with dimension  $k$  for some large  $q$ ; and a smaller inner binary linear code  $\mathcal{C}_{\text{in}} \subseteq \mathbb{F}_2^{n_0}$ , with dimension  $k_0 = \log_2 q$ . We define the concatenated code  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}} \subseteq \mathbb{F}_2^{n_0 \cdot n}$  by first encoding a message  $m \in \mathbb{F}_q^k$  (which can also be thought of as  $m \in \mathbb{F}_2^{k_0 \cdot k}$ ) with  $\mathcal{C}_{\text{out}}$ . Then, we encode each symbol of the resulting codeword using  $\mathcal{C}_{\text{in}}$ . That is, for a message  $m$ ,

$$\mathcal{C}(m) = (\mathcal{C}_{\text{in}}(\mathcal{C}_{\text{out}}(m)_1), \mathcal{C}_{\text{in}}(\mathcal{C}_{\text{out}}(m)_2), \dots, \mathcal{C}_{\text{in}}(\mathcal{C}_{\text{out}}(m)_n)) \in \mathbb{F}_2^{n_0 \cdot n}.$$

It is not hard to see that the rate of  $\mathcal{C}$  is the product of the rates of  $\mathcal{C}_{\text{in}}$  and  $\mathcal{C}_{\text{out}}$ , and that the distance of  $\mathcal{C}$  is *at least* the product of the distances of  $\mathcal{C}_{\text{in}}$  and  $\mathcal{C}_{\text{out}}$ .

The natural approach to constructing a good concatenated code is to choose  $\mathcal{C}_{\text{out}}$  and  $\mathcal{C}_{\text{in}}$  with the best known trade-offs: Since  $\mathcal{C}_{\text{out}}$  is over a large alphabet, we know explicit constructions of codes with optimal rate-distance trade-off<sup>2</sup>; and if  $n_0$  is sufficiently small, we can find a  $\mathcal{C}_{\text{in}}$  on the GV bound either deterministically by brute force or else with low randomness, depending on the size of  $n_0$ .

<sup>2</sup> For codes over large alphabets, the best possible trade-off is the *Singleton bound*, or  $R = 1 - \delta$ . This is achievable, for example, by Reed–Solomon codes.



However, in general this approach will not achieve the GV bound. If we do not assume any additional properties of  $\mathcal{C}_{\text{out}}$  and  $\mathcal{C}_{\text{in}}$ , and simply use the concatenation properties, then setting the parameters so that  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  has distance  $\frac{1-\varepsilon}{2}$ , the rate of  $\mathcal{C}$  will be at most roughly  $\varepsilon^3$ . This is known as the *Zyablov bound* [31] (see also [14]). As we discuss more in Section 1.1, concatenation has been a popular approach to obtain fully explicit codes with good rate-distance trade-offs, but none of these constructions are known to beat the Zyablov bound.

Instead of using a *single* inner code, several works have focused on a related construction originally due to Thommesen [29], which uses multiple inner codes. More precisely, this construction uses i.i.d. random linear inner codes for each coordinate. It can be shown [29] that the resulting code does lie on the GV bound with high probability, and if  $\mathcal{C}_{\text{out}}$  is chosen appropriately there are even efficient decoding algorithms for it [10, 27, 15]. However, this approach relies heavily on the fact that the inner codes are independent, and as a result uses a lot of randomness.

This state of affairs motivates the following question (also asked in the title of this paper):

► **Question 1.** *Are there concatenated linear codes  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  (with a single random linear inner code  $\mathcal{C}_{\text{in}}$ ) that meet the GV bound with high probability over  $\mathcal{C}_{\text{in}}$ ?<sup>3</sup> If so, are there sufficient conditions on  $\mathcal{C}_{\text{out}}$  that will guarantee this?*

In this paper, we show that *yes*, there are concatenated codes that meet the GV bound, and we also give two sufficient conditions on  $\mathcal{C}_{\text{out}}$  for this to hold. Our existential result is non-constructive, but it is our hope that our sufficient conditions will lead to explicit constructions of appropriate  $\mathcal{C}_{\text{out}}$ -s, which would lead to explicit (or at least pseudo-random, depending on the alphabet size of  $\mathcal{C}_{\text{out}}$ ) concatenated codes on the GV bound.

► **Remark 1 (Motivation for Question 1).** Above, we have motivated Question 1 as an avenue towards explicit or pseudo-random binary codes on the GV bound, and indeed this is our original motivation. But we point out that Question 1 is also interesting in its own right. Concatenated codes are a classical construction, going back to the 1960's [9], and have been used in many different settings over the decades. It seems like a fundamental question to understand when these codes can attain the GV bound.

► **Remark 2 (Focus on Linear Codes).** In Question 1 and in this paper, we focus on *linear* codes. This is because if we used, say, a uniformly random non-linear code as the inner code, it would require exponentially more randomness than a random linear inner code, so this does not seem like a hopeful avenue for derandomization. We note however that the question is much easier for non-linear codes. For example, suppose that  $\mathcal{C}_{\text{out}}$  is a Reed–Solomon code of rate  $\varepsilon$  so that each symbol is additionally tagged with its evaluation point: that is, the symbol corresponding to  $\alpha \in \mathbb{F}_q$  is  $(\alpha, f(\alpha)) \in \mathbb{F}_q^2$ . For the inner code, we use a completely random (non-linear) code of rate  $\varepsilon$ . Then since all of the symbols in each outer codeword are different by construction, each codeword is essentially uniformly random, and it is not hard to show that the result is close to the GV bound in the sense that a code of rate  $O(\varepsilon^2)$  will have distance  $1/2 - O(\varepsilon)$  with high probability. This same argument will not work when  $\mathcal{C}_{\text{in}}$  is linear, since the different symbols of codewords of  $\mathcal{C}_{\text{out}}$  will still have  $\mathbb{F}_2$ -linear relationships.

<sup>3</sup> Of course, if the length of either the inner code or the outer code is 1, this question reduces to the non-concatenated setting; we are interested in parameter regimes where  $n_0$  is non-trivial.

### Our Contributions

Our main results are:

1. **Existence of concatenated codes on the GV bound.** We answer the first part of Question 1: there *are* concatenated codes  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  that achieve the GV bound, in a wide variety of parameter regimes. In particular, we show that *most* codes  $\mathcal{C}_{\text{out}}$  are actually good:

► **Theorem 3** (Informal; Theorem 4.2 in the full version). *Suppose that  $\mathcal{C}_{\text{out}} \subseteq \mathbb{F}_q^n$  and  $\mathcal{C}_{\text{in}} \subseteq \mathbb{F}_2^{n_0}$  are random linear codes of rate  $\varepsilon$ , so that  $q \geq 2^{\Omega(\varepsilon^{-3})}$ . Then  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  has rate  $\varepsilon^2$ , and with high probability, the relative distance of  $\mathcal{C}$  is at least  $1/2 - O(\varepsilon)$ .*

While Theorem 3 seems intuitive (in the sense that a random linear code lies on the GV bound with high probability, so why not concatenated random linear codes?), to the best of our knowledge it has not appeared in the literature before, and the proof was not obvious (to us).<sup>4</sup> One challenge is that a codeword  $c \in \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  is not uniformly random in  $\mathbb{F}_2^N$ . In particular, the natural strategy of “show that each non-zero codeword has high weight with high probability and union bound” that is used to establish the Gilbert–Varshamov bound will not work in this setting, as we do not have enough concentration.

2. **Sufficient conditions for  $\mathcal{C}_{\text{out}}$ .** Our existence result above uses a random linear code as the outer code, which does not help in the quest for explicit constructions. However, our proof techniques inspire two sufficient conditions on  $\mathcal{C}_{\text{out}}$ . That is, if  $\mathcal{C}_{\text{out}}$  satisfies these conditions, then  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  will meet the GV bound with high probability when  $\mathcal{C}_{\text{in}}$  is a random linear code. Our hope is that formalizing these will lead to explicit constructions in the future.

We give an overview and intuition for our two sufficient conditions here. We note that both conditions are only sufficient when the alphabet size  $q$  for  $\mathcal{C}_{\text{out}}$  is suitably large (exponential in  $1/\text{poly}(\varepsilon)$ ); see Theorems 5.1 and 6.2 in the full version for details.

- **Sufficient Condition 1: A soft-decoding-like condition on  $\mathcal{C}_{\text{out}}^\perp$ .** Our first sufficient condition, formalized in Theorem 5.1 in the full version, is a soft-list-decoding-like condition on  $\mathcal{C}_{\text{out}}^\perp$ . More precisely, we define a distribution  $\mathcal{D}^5$  on the alphabet  $\mathbb{F}_q$ ; the condition is that

$$\Pr_{x \sim \mathcal{D}^n} [x \in \mathcal{C}_{\text{out}}^\perp \setminus \{0\}] \leq \frac{1}{q^k} (1 + \Delta) \quad (1)$$

for some small  $\Delta$ . Note that  $1/q^k$  is the probability that a completely random vector is in  $\mathcal{C}_{\text{out}}^\perp$ , so this condition is saying that if the coordinates of  $x$  are drawn i.i.d. from the same distribution  $\mathcal{D}$ , then  $x$  not much more likely to be in  $\mathcal{C}^\perp$  than in a uniformly random vector. We show that if this holds, then  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  lies on the GV bound with high probability over the choice of a random linear inner code  $\mathcal{C}_{\text{in}}$ .

It’s not hard to see (Remark 8 in the full version) that this condition holds in expectation for a random linear code  $\mathcal{C}_{\text{out}}$ , and in particular there exist linear codes  $\mathcal{C}_{\text{out}}$  that have this property.

<sup>4</sup> We note that earlier work by Barg, Justesen and Thomessen [4] also addresses random linear outer codes concatenated with an arbitrary (fixed) inner code, using very different techniques than we do. They do not explicitly state a statement like Theorem 3 above, though it is plausible that their techniques could be used to prove something similar. We discuss their techniques and the relationship to our work in Section 1.1.

<sup>5</sup> The distribution  $\mathcal{D}$  is intuitively defined as follows. Let  $\mathcal{C}_{\text{in}}$  be the inner code, and suppose that it has a generator matrix  $G_0 \in \mathbb{F}_2^{n_0 \times k_0}$ . Then to sample from  $\mathcal{D}$ , we take a random sparse linear combination of the rows of  $G_0$  (over  $\mathbb{F}_2$ ), and interpret the result in  $\mathbb{F}_2^{k_0}$  as an element of  $\mathbb{F}_q$ , which we return.

This condition is reminiscent of  $\mathcal{C}_{\text{out}}^\perp$  being list-decodable from soft information (e.g., [20]). In soft-list-decoding, one typically gets a distribution  $\mathcal{D}_i$  for each  $i \in [n]$ , interpreted as giving “soft information” about the  $i$ 'th symbol. If one can show that a vector drawn from  $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$  is unlikely to be in the code, this implies that there are not too many codewords that are likely given the soft information we have received. However, there are several differences between existing work on soft list-decoding and our work, notably that our distribution  $\mathcal{D}$  is a particular one and is the same for all  $i$ , and also there are some differences in the parameter settings.

This condition can also be seen as a soft form of *list-recovery*, where we have the same list in each coordinate.<sup>6</sup> In more detail, if the support of  $\mathcal{D}$  is concentrated on a small set  $S$  (which ours is for reasonable settings of  $n_0, \varepsilon$ , see Remark 7 in the full version), then the condition in Theorem 5.1 is related to asking that the number of codewords that lie in the combinatorial rectangle given by  $S \times S \times \cdots \times S$  is about what it should be. Unfortunately, the definition of “small” here does not seem to be small enough for existing constructions of list-recoverable codes (for example folded RS codes or multiplicity codes) to yield any results.

- **Sufficient Condition 2:  $\mathcal{C}_{\text{out}}$  has good min-entropy.** Our second sufficient condition, formalized in Theorem 6.2, requires the codewords of  $\mathcal{C}_{\text{out}}$  to be “smooth”, meaning, roughly, that every nonzero codeword has a fairly uniform distribution of symbols from  $\mathbb{F}_q$ . To illustrate why a smoothness condition is desirable, let us consider two extreme cases.

The bad extreme is when there exists a codeword  $c$  that is supported on very few symbols, say even on a single symbol. If  $c = (\sigma, \sigma, \dots, \sigma)$  for some  $\sigma \in \mathbb{F}_q$ , then the relative weight of  $c \circ \mathcal{C}_{\text{in}}$ , for a random binary inner code  $\mathcal{C}_{\text{in}}$  of rate  $\varepsilon$ , might be  $\frac{1}{2} - \Omega(\sqrt{\varepsilon})$ , much worse than the  $\frac{1}{2} - O(\varepsilon)$  that we would want for the GV bound.

The good (possibly unrealistic) extreme is where each nonzero codeword of  $\mathcal{C}_{\text{out}}$  has a symbol distribution that is *uniform* over  $\mathbb{F}_q$ . In this case it is not hard to see that  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  will be close to the GV bound with high probability over a random linear code  $\mathcal{C}_{\text{in}}$ . (For this, all we need is that  $\mathcal{C}_{\text{in}}$  has about the “right” weight distribution, which a random linear code will have with high probability).

The natural question is thus *how smooth* the codewords of  $\mathcal{C}_{\text{out}}$  should be in order for  $\mathcal{C}$  to have distance  $\frac{1}{2} - O(\varepsilon)$ . In Section 6 in the full version, we quantify this by the *smooth min-entropy* of the codewords’ empirical distributions on symbols. We show in Theorem 6.2 that if this smooth min-entropy is large enough for all  $c \in \mathcal{C}_{\text{out}}$ , then  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  is likely to lie near the GV bound when  $\mathcal{C}_{\text{in}}$  is a random linear binary code.

How large is “large enough”? For this informal discussion, we give one example of the parameter settings from Theorem 6.2: It is enough for every non-zero codeword  $c \in \mathcal{C}_{\text{out}}$  to have a symbol distribution that has  $\Theta(\varepsilon n)$  copies of the same symbol (say, the zero symbol), while the remaining symbols in  $c$  are uniformly distributed over a set of size only  $q^{1-\varepsilon}$ . By some metrics this is still a fairly “spiky” distribution, but it is “smooth enough” for our purposes.

Note that while our soft-decoding-like condition considers  $\mathcal{C}_{\text{out}}^\perp$ , our smooth min-entropy condition here considers  $\mathcal{C}_{\text{out}}$  itself.

<sup>6</sup> Informally, a code  $\mathcal{C} \subseteq \Sigma^n$  is said to be list-recoverable if for any small sets  $S_1, \dots, S_n \subseteq \Sigma$ , there are not too many codewords  $c \in \mathcal{C}$  so that  $c_i \in S_i$  for many values of  $i$ .

## 1.1 Related Work

### Explicit Concatenated Codes

Concatenation (with a single inner code) has been a common approach to obtain explicit codes close to the GV bound. Here we mention a few such places this comes up. Choosing  $\mathcal{C}_{\text{out}}$  to be the Reed–Solomon code, and  $\mathcal{C}_{\text{in}}$  to be the Hadamard code, gets a code of length  $O(k^2/\varepsilon^2)$  for any dimension  $k$  [3], and replacing Reed–Solomon with the Hermitian code gets length  $O((k/\varepsilon)^{5/4})$  [6]. Choosing a different AG code for  $\mathcal{C}_{\text{out}}$  can result in non-vanishing rate and in fact approach rate  $\varepsilon^3$  (see [28]). Moreover, concatenating Reed–Solomon with the Wozencraft ensemble gives the *Justesen* code [19], having constant relative rate and constant relative distance. Note that none of these concatenation-based constructions thus far have beat the Zyablov bound.

### Concatenated Codes with Random Linear $\mathcal{C}_{\text{out}}$

Relevant to Theorem 3, [4] studies a random linear code  $\mathcal{C}_{\text{out}}$  concatenated with a *fixed* inner code  $\mathcal{C}_{\text{in}}$ . (See also [5], which applies the same techniques for an application in compressive sensing). The work [4] derives bounds on the distance of  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  in terms of (moments of) the weight distribution of  $\mathcal{C}_{\text{in}}$ . These bounds imply that  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  approaches the GV bound in some cases, but doesn't seem to immediately imply Theorem 3.

Before discussing their techniques more, we note that the biggest difference between [4] and our work is that their question is about the behavior of random linear codes, and so naturally their approach crucially uses the fact that  $\mathcal{C}_{\text{out}}$  is random. In contrast, the motivation for our work is to find deterministic sufficient conditions on  $\mathcal{C}_{\text{out}}$ , and we invoke a random linear outer code as a proof of concept that our approach is realizable.

Next, we briefly describe the techniques and implications of [4], relative to Theorem 3. The key result of [4] is an expression of the limiting trade-off between the rate  $R$  and the distance  $\delta$  of  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$ , in terms of the function  $\phi(\tau) = \ln \mathbb{E}_X[e^{\tau X}]$ , where  $X$  is the weight of a random codeword from  $\mathcal{C}_{\text{in}}$  and where  $\tau \leq 0$  parameterizes the trade-off.<sup>7</sup> They show that this trade-off meets the GV bound when  $\mathcal{C}_{\text{in}}$  is the identity (trivial) code, and investigate how it behaves when  $\mathcal{C}_{\text{in}}$  is a non-trivial code. Towards this, one can use their trade-off to work out the Taylor series for  $R$  around  $\delta = 1/2$ . It is not hard to see that under mild conditions on  $\mathcal{C}_{\text{in}}$ , the first two terms of this Taylor expansion vanish and hence we obtain  $R = \Theta(\varepsilon^2) + O_{\mathcal{C}_{\text{in}}}(\varepsilon^3)$  when  $\delta = 1/2 - \varepsilon$ , where the  $O_{\mathcal{C}_{\text{in}}}(\cdot)$  notation hides constants that depend on  $\mathcal{C}_{\text{in}}$ . This implies that if  $n_0$  is a constant, independent even of  $\varepsilon$ , then  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  approaches the GV bound. However, if  $n_0$  is growing relative to  $\varepsilon$  (which it is in our case, as we take  $\mathcal{C}_{\text{in}}$  to have rate  $\varepsilon$ ), then the “constant” terms hiding in the  $O_{\mathcal{C}_{\text{in}}}(\varepsilon^3)$  term may depend on  $n_0$ , which in turn may depend on  $\varepsilon$ . It seems plausible that when  $\mathcal{C}_{\text{in}}$  is a random linear code, this dependence is mild<sup>8</sup> and something like Theorem 3 could be established with these techniques, but to the best of our knowledge such a proof has not appeared in the literature and does not seem to follow immediately.

<sup>7</sup> In more detail, this trade-off is given by  $R = \frac{1}{n_0 \ln(2)}(\tau\phi'(\tau) - \phi(\tau))$  and  $\delta = \frac{\phi'(\tau)}{n_0}$ , for  $\tau \leq 0$ .

<sup>8</sup> In particular, as pointed out in [4], the first  $d^\perp - 1$  terms of the Taylor series will agree with the GV bound, where  $d^\perp$  is the dual distance of  $\mathcal{C}_{\text{in}}$ , which for a random linear code  $\mathcal{C}_{\text{in}}$  is quite large.

### Non-Concatenation-Based Explicit Constructions

As mentioned above, there have been several breakthroughs in the past few years obtaining explicit constructions of binary codes near the GV bound, and even efficient algorithms for them. In a breakthrough result, Ta-Shma [28] constructed *explicit* linear codes of relative distance  $\frac{1-\varepsilon}{2}$  having rate  $\varepsilon^{2+o(1)}$ . Ta-Shma's codes are also  $\varepsilon$ -balanced, i.e.,  $\Delta(x, y) \in [\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}]$ , and thus give rise to explicit  $\varepsilon$ -biased sample spaces, which are ubiquitous in pseudorandomness and derandomization. Works that followed gave efficient *decoding* of Ta-Shma codes and their variants [1, 16, 17, 26, 18] (see also [7] for a different, randomized, construction that slightly improves upon the rate of [28], and admits efficient decoding). We note that these codes are graph-based, and do not in general have a concatenated structure.

### Results with Multiple i.i.d. Inner Codes

Thommesen showed that when the outer code is a Reed–Solomon code, and it is concatenated with  $n$  different random linear codes, one for each coordinate, chosen independently, then the resulting code lies on the GV bound with high probability [29]. Guruswami and Indyk devised efficient decoding algorithms for these codes, based on *list-recoverability* of the outer code [10]. That work used a Reed–Solomon code as the outer code, which is list-recoverable up to the Johnson bound. Later, Rudra [27] observed that the parameters could be improved by swapping out the Reed–Solomon code for a code that can be list-recovered up to capacity, for example a Folded Reed–Solomon code. Later work obtained nearly-linear-time decoding algorithms by swapping out the outer code for a capacity-achieving list-recoverable code with near-linear-time list-recovery algorithms [15, 21]. Codes with multiple i.i.d. inner codes have also been studied in [32, 8].

We also mention the work of Guruswami and Rudra [13], who show that the same construction (a list-recoverable code concatenated with  $n$  different i.i.d. random linear codes) is *list-decodable* up to capacity with high probability. In the results [10, 27, 15, 21] mentioned above, list-recovery of the outer code was needed for *algorithms*, not the combinatorial result (which follows already from [29]). In contrast, in [13], the list-recoverability of the outer code is needed for the combinatorial result itself. In that sense, the flavor is similar to our sufficient condition in Section 5 in the full version, although the techniques are very different, and in our work we only use one inner code.

### Further Low-randomness Constructions of Binary Codes on GV Bound

If one's goal is to explicitly construct a binary code that achieves that GV bound, at least two types of partial results may be considered as subgoals. In the first class of results, one seeks explicit codes whose rate vs. distance tradeoff is as close to the GV bound as possible. This includes the works discussed in the first two paragraphs of Section 1.1 above. A second path is to seek codes that fully attain the GV bound, and strive to minimize the amount of randomness used in their construction.

Varshamov's classic result [30] is that a random linear code likely achieves the GV bound. Constructing such a code of length  $n$  and rate  $R$  requires sampling either a random generating matrix or a random parity-check matrix, and thus  $O(\min\{R, 1 - R\} \cdot n^2)$  random bits are needed. Two classical elementary constructions – the Wozencraft ensemble [22] and the random Toeplitz Matrix construction (e.g., [14, Exercise 4.6]) – are able to reduce the needed randomness to  $O(n)$ .

So far, no codes achieving the GV bound using  $o(n)$  randomness are known. Moreover, there is a certain natural obstacle, which we now describe, that needs to be tackled before sublinear randomness can be achieved. Say that a random code  $\mathcal{C} \subseteq \mathbb{F}_2^n$  is *uniform* if

every  $x \in \mathbb{F}_2^n \setminus \{0\}$  appears in the code with the same probability, namely,  $p_{R,n} = \frac{2^{Rn}-1}{2^n-1}$ . It is not hard to prove via a union bound that a uniform linear code achieves the GV bound with high probability (this is exactly Varshamov's observation). To the best of our knowledge, every known GV-bound construction to date, including the linear randomness constructions mentioned above, is uniform. Unfortunately, a uniform code ensemble with sublinear randomness cannot exist as long as  $R$  is bounded away from 1. Indeed, to have events that occur with probability  $p_{R,n}$ , at least  $\log_2 \frac{1}{p_{R,n}} \approx (1-R)n$  random bits are required. Therefore, a code construction obtaining the GV bound with sublinear randomness would have to do so without being uniform (see also [23, Section 5]). We have hope that our sufficient conditions in Theorems 5.1 and 6.2 could be attained by non-uniform codes. For example, as discussed above, the soft-decoding-like condition of Theorem 5.1 is reminiscent of results on soft-list-decoding and soft-list-recovery, which in different parameter regimes can even be achieved by deterministic codes.

A related line of work [12, 25, 23] attempts to construct codes that enjoy a broad class of desirable combinatorial properties similar to those of random linear codes using as little randomness as possible. Such properties include not just the GV bound, but also list decodability up to the *Elias bound* (see [23]), list recoverability, and, more generally, *local similarity* (see [23, Definition 2.14]) to a random linear code.

## 1.2 Technical Overview

In this section we give an overview of the main technical ideas. This section also serves as an outline of the full version of the paper.

### Section 3: A moment-based framework

In Section 3, we set up a framework that will be useful for the results in Section 4 and Section 5. We describe this approach here.

Suppose that we are trying to encode a message  $m \in \mathbb{F}_q^k$  with our concatenated code  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$ , to obtain  $\mathcal{C}(m) = w \in \mathbb{F}_2^{n \cdot n_0}$ . Each symbol of  $w$  is indexed by some  $\alpha \in [n]$  and some  $\beta \in [n_0]$ ; this symbol is equal to

$$(\mathcal{C}_{\text{in}}(\mathcal{C}_{\text{out}}(m)_\alpha))_\beta = \langle \mathcal{C}_{\text{out}}(m)_\alpha, b_\beta \rangle,$$

where  $b_\beta$  is the  $\beta$ 'th row for a generator matrix  $G_0 \in \mathbb{F}_2^{n_0 \times k_0}$  for  $\mathcal{C}_{\text{in}}$ , and where the  $\langle \cdot, \cdot \rangle$  notation denotes the dot product over  $\mathbb{F}_2$ . This motivates the definition of a variable  $X_m \in \mathbb{R}$  defined by

$$X_m = \sum_{\alpha \in [n]} \sum_{\beta \in [n_0]} (-1)^{\langle \mathcal{C}_{\text{out}}(m)_\alpha, b_\beta \rangle}.$$

Indeed,  $X_m$  is the bias of  $w = \mathcal{C}(m)$ ; the weight of  $w$  is at least  $\frac{1}{2} - O(\varepsilon N)$  if and only if  $X_m$  is at most  $O(\varepsilon N)$ . Thus, to show that the code  $\mathcal{C}$  has distance at least  $\frac{1}{2} - O(\varepsilon N)$ , it suffices to show that

$$\max_{m \in \mathbb{F}_q^k \setminus \{0\}} X_m = O(\varepsilon N).$$

Our strategy will be to consider a large moment of  $X_m$  over the choice of a random nonzero message  $m$ :

$$\mathbb{E}_{m \sim \mathbb{F}_q^k \setminus \{0\}} [X_m^r]$$

for some appropriate  $r$ . If we can show that this is smaller than  $(c\varepsilon N)^r/q^k$ , then Markov’s inequality will imply that

$$\Pr_{m \sim \mathbb{F}_q^k \setminus \{0\}} [X_m \geq c\varepsilon N] \leq \frac{\mathbb{E}_{m \sim \mathbb{F}_q^k \setminus \{0\}} [X_m^r]}{(c\varepsilon N)^r} < \frac{1}{q^k},$$

and in particular that there are no messages  $m$  so that  $X_m \geq c\varepsilon N$ .

In Lemma 3.3, we take a Fourier transform in order to re-write  $\mathbb{E}[X_m^r]$  as a quantity involving  $\mathcal{C}_{\text{out}}^\perp$ . This quantity can be thought of as follows. For every integer-valued matrix<sup>9</sup>  $V \in \mathbb{Z}_{\geq 0}^{n_0 \times n}$  with entries that sum to  $r$ , we consider a vector  $g_V \in \mathbb{F}_q^n$  defined by considering the matrix  $G_0^T \cdot V \in \mathbb{F}_2^{k_0 \times n}$  and then treating it as a vector  $g_V \in \mathbb{F}_q^n$  by identifying each of the columns in  $\mathbb{F}_2^{k_0}$  with elements of  $\mathbb{F}_q$ . Then the quantity in Lemma 3.3 has to do with the number of these vectors  $g_V$  that are in  $\mathcal{C}_{\text{out}}^\perp$ . The exact expression doesn’t matter too much for this informal discussion; instead we explain below how we use this re-writing to prove Theorem 4.2 and Theorem 5.1.

#### Section 4: Most codes $\mathcal{C}_{\text{out}}$ are good

Theorem 4.2 informally says that if  $\mathcal{C}_{\text{out}}$  is a random linear code, then with high probability  $\mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  is near the GV bound. In the proof, we use our framework from Section 3, and show that with high probability over  $\mathcal{C}_{\text{out}}$ , the moment  $\mathbb{E}_m[X_m^r]$  is small for an appropriate  $r$ . To do this, we need to count the number of matrices  $V$  described above that are likely to land in  $\mathcal{C}_{\text{out}}^\perp$ . Since  $\mathcal{C}_{\text{out}}$  is a random linear code, so is  $\mathcal{C}_{\text{out}}^\perp$ , and so the probability of any particular *non-zero*  $g_V$  landing in it is small (about  $1/q^k$ ), while of course the probability that 0 is contained in  $\mathcal{C}_{\text{out}}^\perp$  is 1. Thus, the challenge is understanding how many  $g_V$ -s are actually zero. There are two ways that a matrix  $V$  as described above could lead to  $g_V = 0$ : Either  $V = 0 \pmod 2$ , or else  $V$  is non-zero mod 2 but  $G_0^T V = 0$ . The first case can be counted straightforwardly. For the second, we leverage the weight distribution that the inner code  $\mathcal{C}_{\text{in}}$  is likely to have. We note that this is the only place (in any of our arguments) that we need  $\mathcal{C}_{\text{in}}$  to be a random linear code: We just need it to have approximately the “right” weight distribution.

#### Section 5: A soft-decoding-like sufficient condition

The expression that we get for  $\mathbb{E}_m[X_m^r]$  in Lemma 3.3 directly inspires our soft-decoding-like sufficient condition in Theorem 5.1. One can view the task of counting the matrices  $V$  so that  $g_V \in \mathcal{C}_{\text{out}}^\perp$  as choosing a random  $V$  and asking about the probability that  $g_V \in \mathcal{C}_{\text{out}}^\perp$ . If the columns of  $V$  were independent, then this would be the same as choosing the coordinates of  $g_V$  i.i.d. from some distribution  $\mathcal{D}$ . Thus we would get a requirement on  $\Pr_{x \sim \mathcal{D}^n} [x \in \mathcal{C}_{\text{out}}^\perp]$ , similar to the condition in Equation (1) that we end up with.

Of course, the coordinates are not independent (because the total weight of  $V$  is fixed to be  $r$ ), but this can be solved. In more detail, we choose  $r$  to be a Poisson random variable, which in this setting makes the columns of  $V$  independent. One hiccup is that the “Poisson-ized” distribution turns out to be meaningfully different than the original distribution, in the sense that it is much more likely that  $g_V = 0$  in the Poisson-ized version. This means that the “natural” soft-decoding-like condition that one would get out of this is not realizable: The probability that  $g_V \in \mathcal{C}_{\text{out}}^\perp$  is much bigger than we want it to be, for *any*

<sup>9</sup> In the actual quantity, the entries of this matrix are ordered, and we denote it  $\mathcal{V}$  instead of  $V$ ; we ignore the ordering in this discussion for simplicity.

$\mathcal{C}_{\text{out}}$ , just because  $g_V$  is too likely to be zero. Fortunately, this seems to be the only obstacle: as in Equation (1), we separate out the  $g_V = 0$  term (using the analysis from Section 4) to arrive at a condition that *is* realizable. We explain why the condition is realizable – that is, why there exists a  $\mathcal{C}_{\text{out}}$  that meets it – in Remark 8.

### Section 6: A smoothness condition on $\mathcal{C}_{\text{out}}$

For our second sufficient condition, we depart from our moment-based framework and work from first principles. Our main theorem in Section 6 is Theorem 6.2, which informally says that if the elements of  $\mathcal{C}_{\text{out}}$  have “smooth” enough distributions of symbols, in the sense that they each have large enough min-entropy, that  $\mathcal{C} = \mathcal{C}_{\text{out}} \circ \mathcal{C}_{\text{in}}$  will lie near the GV bound with high probability. The basic idea is to consider a *worst-case* assignment of symbols in  $\mathbb{F}_q$  to codewords in  $\mathcal{C}_{\text{in}}$ ; this assignment need not be linear and can depend on a particular codeword  $c \in \mathcal{C}_{\text{out}}$ . Such a worst-case assignment would simply assign the lowest-weight codewords in  $\mathcal{C}_{\text{in}}$  to the most frequent symbols in a codeword  $c \in \mathcal{C}_{\text{out}}$ . Using the weight distribution that  $\mathcal{C}_{\text{in}}$  is likely to have, along with the min-entropy assumption, we can show that this worst-case assignment will *still* result in codewords  $w \in \mathcal{C}$  of weight at least  $\frac{1}{2} - O(\varepsilon)$ .

We note that, unlike our sufficient condition from Section 5, we don’t have a proof of feasibility for our smoothness condition. That is, as far as we know, there may not be any linear code  $\mathcal{C}_{\text{out}}$  that is smooth in this sense. However, as a proof of concept we mention in Remark 9 that a random linear code will have a similar property with high probability. Moreover, we find it plausible that codewords of *algebraically structured* codes (say, Folded Reed–Solomon codes, Folded Multiplicity, or even large sub-codes of plain Reed–Solomon codes), would satisfy this property, even if a random code does not.

---

### References

- 1 Vedat Levi Alev, Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. List decoding of direct sum codes. In *Proceedings of the 31st Symposium on Discrete Algorithms (SODA 2020)*, pages 1412–1425. ACM-SIAM, 2020.
- 2 Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *Information Theory, IEEE Transactions on*, 38(2):509–516, 1992.
- 3 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost  $k$ -wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
- 4 Alexander Barg, Jørn Justesen, and Christian Thommesen. Concatenated codes with fixed inner code and random outer code. *IEEE Transactions on Information Theory*, 47(1):361–365, 2001.
- 5 Alexander Barg and Arya Mazumdar. Small ensembles of sampling matrices constructed from coding theory. In *2010 IEEE International Symposium on Information Theory*, pages 1963–1967. IEEE, 2010.
- 6 Avraham Ben-Aroya and Amnon Ta-Shma. Constructing small-bias sets from algebraic-geometric codes. *Theory of Computing*, 9(5):253–272, 2013.
- 7 Guy Blanc and Dean Doron. New near-linear time decodable codes closer to the GV bound. In *Proceedings of the 37th Computational Complexity Conference (CCC 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.
- 8 È L Blokh and Victor Vasilievich Zyablov. Existence of linear concatenated binary codes with optimal correcting properties. *Problemy Peredachi Informatsii*, 9(4):3–10, 1973.
- 9 G. David Forney. Concatenated codes. Technical Report 440, Research Laboratory of Electronics, MIT, 1965.





- 10 Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *Proceedings of the 15th Symposium on Discrete Algorithms (SODA 2004)*, pages 756–757. ACM-SIAM, 2004.
- 11 Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.
- 12 Venkatesan Guruswami and Jonathan Mosheiff. Punctured Low-Bias Codes Behave Like Random Linear Codes. In *Proceedings of the 63rd Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 36–45. IEEE, 2022.
- 13 Venkatesan Guruswami and Atri Rudra. The existence of concatenated codes list-decodable up to the hamming bound. *IEEE Transactions on information theory*, 56(10):5195–5206, 2010.
- 14 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. URL: <http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book>.
- 15 Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *SIAM Journal on Computing*, pages FOCS17–157, 2019.
- 16 Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit  $\varepsilon$ -balanced codes near the Gilbert–Varshamov bound. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 434–445. IEEE, 2020.
- 17 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 Symposium on Theory of Computing (STOC 2021)*, pages 1527–1536. ACM, 2021.
- 18 Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. List decoding of tanner and expander amplified codes from distance certificates. In *Proceedings of the 64th Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 1682–1693. IEEE, 2023.
- 19 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18(5):652–656, 1972.
- 20 Ralf Koetter and Alexander Vardy. Algebraic soft-decision decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.
- 21 Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *IEEE Transactions on Information Theory*, 67(1):296–316, 2020.
- 22 James L. Massey. Threshold decoding. Technical Report 410, Research Laboratory of Electronics, MIT, 1963.
- 23 Jonathan Mosheiff, Nicolas Resch, Kuo Shang, and Chen Yuan. Randomness-efficient constructions of capacity-achieving list-decodable codes. *arXiv preprint*, 2024.
- 24 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.
- 25 Aaron (Louie) Putterman and Edward Pyne. Pseudorandom Linear Codes Are List-Decodable to Capacity. In *Proceedings of the 15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, pages 90:1–90:21. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.
- 26 Silas Richelson and Sourya Roy. Gilbert and Varshamov meet Johnson: List-decoding explicit nearly-optimal binary codes. In *Proceedings of the 64th Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 194–205. IEEE, 2023.
- 27 Atri Rudra. *List decoding and property testing of error-correcting codes*. University of Washington, 2007.
- 28 Amnon Ta-Shma. Explicit, almost optimal,  $\varepsilon$ -balanced codes. In *Proceedings of the 49th Annual Symposium on Theory of Computing (STOC 2017)*, pages 238–251. ACM, 2017.
- 29 Christian Thommesen. The existence of binary linear concatenated codes with Reed–Solomon outer codes which asymptotically meet the Gilbert–Varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, 1983.

## 53:12 When Do Concatenated Codes Approach The GV Bound?

- 30 Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Doklady Akad. Nauk, SSSR*, 117:739–741, 1957.
- 31 Victor Vasilievich Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Problemy Peredachi Informatsii*, 7(1):5–13, 1971.
- 32 Victor Vasilievich Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Problemy Peredachi Informatsii*, 7(1):5–13, 1971.

# Parallel Repetition of $k$ -Player Projection Games

Amey Bhangale  

Department of Computer Science and Engineering, University of California, Riverside, CA, USA

Mark Braverman  

Department of Computer Science, Princeton University, NJ, USA

Subhash Khot  

Department of Computer Science, Courant Institute of Mathematical Sciences,  
New York University, NY, USA

Yang P. Liu  

School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA

Dor Minzer  

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

We study parallel repetition of  $k$ -player games where the constraints satisfy the *projection* property. We prove exponential decay in the value of a parallel repetition of projection games with a value less than 1.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Interactive proof systems

**Keywords and phrases** Parallel Repetition, Multiplayer games, Projection games

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.54

**Category** RANDOM

**Funding** *Amey Bhangale*: Supported by the Hellman Fellowship award.

*Mark Braverman*: Supported by the National Science Foundation under the Alan T. Waterman Award, Grant No. 1933331.

*Subhash Khot*: Supported by the NSF Award CCF-2130816 and the Simons Investigator Award.

*Yang P. Liu*: Partially supported by NSF DMS-1926686.

*Dor Minzer*: Supported by NSF CCF award 2227876 and NSF CAREER award 2239160.

**Acknowledgements** We thank Kunal Mittal for helpful discussions at the early stage of this work. We also thank anonymous reviewers for helpful suggestions

## 1 Introduction

We study  $k$ -player one-round games and the effect on the value of the game when we repeat the game in parallel.

In a  $k$ -player game  $G$ , a verifier chooses  $k$  questions  $(x^1, x^2, \dots, x^k)$  from a distribution  $\mu$  on the set of questions  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$  and sends  $x^i$  to player  $i$ . Player  $i$  responds to the verifier's question by sending an answer  $a^i \in \mathcal{A}_i$  without communicating with the other players. The verifier accepts the answers based on a fixed predicate  $V((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$ . The value of the game, denoted by  $\text{val}(G)$ , is the maximum, over the players' strategies, accepting probability of the verifier.

The  $n$ -fold parallel repetition of  $G$ , denoted by  $G^{\otimes n}$ , is defined as follows. The verifier sends questions  $\vec{x}^j = (x_1^j, x_2^j, \dots, x_n^j)$  to the  $k$  players where for each  $j \in [n]$ ,  $(x_1^j, x_2^j, \dots, x_n^j)$  is sampled from the original distribution  $\mu$  independently. The  $i^{\text{th}}$  player responds with answers  $\vec{a}^i \in \mathcal{A}_i^n$ . The verifier accepts the answers iff  $V((x_1^j, x_2^j, \dots, x_n^j), (a_1^j, a_2^j, \dots, a_n^j)) = 1$  for each  $j \in [n]$ .



© Amey Bhangale, Mark Braverman, Subhash Khot, Yang P. Liu, and Dor Minzer;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques  
(APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 54; pp. 54:1–54:16



Leibniz International Proceedings in Informatics

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

If  $\text{val}(G) = 1$ , then it is easy to observe that  $\text{val}(G^{\otimes n})$  is also 1. Also,  $\text{val}(G^{\otimes n}) \geq \text{val}(G)^n$  as the players can achieve value  $\text{val}(G)^n$  in the game  $G^{\otimes n}$  by simply repeating an optimal strategy for the game  $G$  independently in all the  $n$  coordinates. The question of interest is how does the quantity  $\text{val}(G^{\otimes n})$  decay with  $n$  if the value of the game  $G$  is less than 1?

Verbitsky [30] showed that for any  $k$ -player game  $G$ , if  $\text{val}(G) < 1$ , then  $\text{val}(G^{\otimes n}) \leq \frac{1}{\alpha(n)}$  where  $\alpha(n)$  is an inverse Ackermann function. This result uses the Density Hales-Jewett Theorem [15, 27] as a black box. For 2-player games, Raz [29] showed that if  $\text{val}(G) < 1$ , then  $\text{val}(G^{\otimes n}) \leq 2^{-\Omega_G(n)}$ , where we use  $\Omega_G(\cdot)$  to clarify that the constant depends on the game  $G$ . There have been many improvements that improve the constants in the bounds, and even get better bounds based on the value  $\text{val}(G)$  of the initial game [21, 28, 12, 7]. These results on parallel repetition of 2-player games have found many applications in probabilistically checkable proofs and hardness of approximation [4, 13, 23].

Mittal and Raz [26] showed that a strong parallel repetition theorem (i.e., the value of  $G^{\otimes n}$  decays exponentially in  $n$  in a certain strong sense) for a particular class of more than 2-player games implies super-linear lower bounds for Turing machines in the non-uniform model. For any  $k \geq 2$ , Dinur, Harsha, Venkat, and Yuen [10] showed that for a large class of  $k$ -player games, called the *connected games*, the exponential decay indeed holds. The class of connected games is defined as follows: define the graph  $H_G$ , whose vertices are the ordered  $k$ -tuples of questions to the  $k$ -players, and there is an edge between questions  $q$  and  $q'$  if they differ in the question to exactly one of the  $k$  players, and are the same for the remaining  $k - 1$  players. The game is said to be connected if the graph  $H_G$  is connected.

A special 3-player (non-connected) game, called the GHZ Game [10], has received much attention. The GHZ game, first introduced by Greenberger, Horne, and Zeilinger [19], is a central game in the study of quantum entanglement. Holmgren and Raz [22] gave the first polynomial decay in the parallel repetition of the GHZ game. Girish, Holmgren, Mittal, Raz, and Zhan [16] later gave a simpler proof of the polynomial decay. Very recently, Braverman, Khot, and Minzer [8], using a much simpler proof, improved these previous results and showed an exponential decay in the GHZ game.

Girish, Holmgren, Mittal, Raz, and Zhan [17] considered the problem of parallel repetition for 3-player games with binary questions and answers and showed polynomial decay for these games. This was later improved by a subset of the authors [18] to all 3-player games over binary questions and *arbitrary* answer lengths. They also study [17] *player-wise connected* games  $G$  that are defined as follows. For each player  $i$ , define the graph  $H_i(G)$ , whose vertices are the possible questions for player  $i$ , and two questions  $x$  and  $x'$  are connected by an edge if there exists a vector  $y$  of questions for all other players, such that both  $(x, y)$  and  $(x', y)$  are asked by the verifier with non-zero probability. The game  $G$  is player-wise connected if, for every  $i$ , the graph  $H_i(G)$  is connected. Girish *et al.* [17] showed polynomial decay in the value of  $n$ -fold parallel repetition of all player-wise connected games. Observe that the notion of player-wise connectedness is more general than the notion of connected games defined above.

In this paper, we will study a special type of  $k$ -player games, that we refer to as projection games. The formal definition is as follows.

► **Definition 1.** For any  $k \geq 2$ , a  $k$ -player game  $G$  is called a *projection game* if for every  $k$ -tuple of question  $q = (x^1, x^2, \dots, x^k)$ , there is  $D_q \geq 1$  and projections  $\sigma_q^i : \mathcal{A}_i \rightarrow [D_q]$  for  $i \in [k]$ , such that  $V((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$  is true iff  $\sigma_q^i(a^i) = \sigma_q^{i'}(a^{i'})$  for any  $i \neq i'$ .

For every question  $q = (x^1, x^2, \dots, x^k)$ , consider a  $k$ -partite hypergraph  $\mathcal{H}_q$  on the vertex set  $(\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_k)$  where  $(a^1, a^2, \dots, a^k)$  is an hyperedge if and only if  $V((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$  is true. Then, the projection property means that for every  $k$ -tuple of

questions  $q$  in the support of  $\mu$ , each connected component in  $\mathcal{H}_q$  is a complete  $k$ -partite hypergraph. Note that this definition of projection games is slightly more general than the usual notion of projection 2-player games [29, 28] where one of the maps  $\sigma_q^i$  (either  $\sigma_q^1$  or  $\sigma_q^2$ ) is an injective map.

Our main theorem shows that if the value of a projection game  $G$  is less than 1, then the value of  $n$ -fold parallel repetition of  $G$  decays exponentially in  $n$ .

► **Theorem 2.** *For any  $k \geq 2$ , a projection  $k$ -player game  $G$  and  $\varepsilon > 0$ , if  $\text{val}(G) = 1 - \varepsilon$ , then  $\text{val}(G^{\otimes n}) \leq \exp(-\Omega_{\varepsilon, G}(n))$ .*

Projection games are a natural subclass of general games. They played a key role in the development [2, 3, 14] of Probabilistically Checkable Proofs (PCPs). In fact, parallel repetition from 2-player projection games had been useful in proving many [20, 25, 24, 9, 11] tight hardness of approximation results, starting with the work of Arora, Babai, Stern, and Sweedyk [1], Bellare, Goldreich, and Sudan [4], and Håstad [23].

Feige [13] used a  $k$ -player projection game, and parallel repetition of the game, to show almost tight hardness of approximating the Set-Cover problem. The decay in the value of a parallel-repeated game, in that case, follows easily from the parallel-repetition theorem for the 2-player game, as the subgame restricted to any two players has a value less than 1.

There are  $k$ -player projection games where the decay in the value of a parallel-repeated game does not trivially follow from the parallel-repetition theorem for the 2-player game. To give a concrete example, consider a simultaneous Max-3-SAT instance problem defined in [6]: the instance consists of  $n$  variables  $X = \{x_1, x_2, \dots, x_n\}$  and  $k$  instances,  $\phi_1, \phi_2, \dots, \phi_k$ , of Max-3-SAT defined over the same set of variables  $X$ . The verifier chooses a variable  $x \in X$  at random and selects clauses  $C_i \in \phi_i$  independently such that  $x \in C_i$  for all  $i \in [k]$ . The verifier sends clause  $C_i$  to player  $i$  and expects a satisfying assignment from  $\{0, 1\}^3$  to  $C_i$  from player  $i$ . The verifier checks if the assignments returned by the players agree on  $x$ . Consider the scenario when it is possible to satisfy any  $(k - 1)$  out of  $k$  instances of Max-3-SAT simultaneously, but there is no assignment to  $X$  that will satisfy all the  $k$  instances. In this case, the value of the game is less than 1. For any  $k'$ -player subgame, where  $k' < k$ , the value of the subgame is 1. Therefore, we cannot use the parallel repetition of 2-player games to conclude that the value of  $n$ -fold parallel repetition of projection games decays with  $n$ .

## 1.1 Proof outline

As mentioned in the introduction, Dinur, Harsha, Venkat, and Yuen [10] showed that for any connected  $k$ -player games  $H$  with  $\text{val}(H) < 1$ , the exponential decay holds for the value of  $H^{\otimes n}$ . We start with a  $k$ -player game  $G$  which is not connected to begin with. At a high level, we transform the game  $G$  to another game  $H$  where  $H$  is connected. While doing such a transformation, we want to make sure we have the following two properties.

1. If  $\text{val}(G) < 1$ , then  $\text{val}(H) < 1$ .
2. There is a way to relate  $\text{val}(G^{\otimes n})$  with  $\text{val}(H^{\otimes n})$ , possibly with a small loss in the constants in the exponent.

As  $H$  is connected, we have  $\text{val}(H^{\otimes n}) = \exp(-\Omega_H(n))$  and this will complete the proof.

There is a trivial transformation that makes any game connected – add all possible  $k$ -tuple of questions, play the game  $G$  on the original questions, and accept all the newly added questions by default. It is easy to see that if  $\text{val}(G) < 1$ , then the value of the transformed game is less than 1. However, in this case, there does not seem to be an easy way to relate  $\text{val}(G^{\otimes n})$  to the value of  $n$ -fold parallel repetition of the transformed game.

## 54:4 Parallel Repetition of k-Player Projection Games

In order to overcome the issue, we make the game  $G$  connected gradually. More concretely, we start with a game  $G_0 = G$  and iteratively, we convert the game  $G_\ell$  to  $G_{\ell+1}$  for  $\ell = 0, 1, \dots$  with the following three properties.

1. For every  $\ell \geq 0$ , the game  $G_\ell$  is a  $k$ -player game with the questions from the set  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$ .
2. The game  $G_{\ell+1}$  is *richer* than the game  $G_\ell$ . In our case, we would be interested in increasing the support of distribution on questions, i.e.,  $\text{supp}(\mu(G_{\ell+1})) \supseteq \text{supp}(\mu(G_\ell))$  (unless, of course,  $\text{supp}(\mu(G_\ell))$  is full).
3. We can relate the value of the game  $G_\ell^{\otimes n}$  to the value of the game  $G_{\ell+1}^{\otimes n}$  up to a fixed polynomial factor. Furthermore,  $\text{val}(G_{\ell+1}) < 1$  if  $\text{val}(G_\ell) < 1$ .

Let us see that this is enough to prove our main theorem. Using properties 1 and 2, for some  $t \geq 1$ , which only depends on the size of the game  $G$ , we can conclude that the game  $G_t$  has full support and hence is *connected*. Using property 3, we have  $\text{val}(G^{\otimes n}) \approx \text{val}(G_t^{\otimes n})^{C_t}$ , where  $C_t > 0$  is a constant that only depends on  $t$ , and furthermore  $\text{val}(G_t) < 1$  if  $\text{val}(G) < 1$  to begin with. Finally, using the result by Dinur, Harsha, Venkat, and Yuen [10] on connected games, we have,  $\text{val}(G_t^{\otimes n}) = \exp(-\Omega_{\varepsilon, G_t}(n))$ , and hence  $\text{val}(G^{\otimes n}) = \exp(-\Omega_{\varepsilon, t, G}(n))$  if  $\text{val}(G) < 1$ .

### The transformation $G_\ell$ to $G_{\ell+1}$

The key idea is to use the *path-trick* from [5] to transform the game  $G_\ell$  (and  $G_\ell^{\otimes n}$ ) to another game  $G_{\ell+1}$  (and  $G_{\ell+1}^{\otimes n}$ ) such that the support of the transformed game is potentially larger than the original game. We illustrate the idea of such a transformation in a 3-player game  $G_\ell$ .

We start with the game  $G_\ell$  and let  $\mu(G_\ell)$  be the distributions on the questions in  $G_\ell$ . For every pair of question-triples  $q = (x, y, z)$  and  $q' = (x', y', z')$  from  $\text{supp}(\mu(G_\ell))$  such that  $x = x'$ , we add a question triple  $\Pi^3((q, q')) := (x, y, z')$  to the game  $G_{\ell+1}$ . Note that in this case, we took two question-triples  $(q, q')$  that share player 1's question and generate a question-triple in the new game with the first two players' questions from  $q$  and player 3's question from  $q'$ . We now state the set of accepting assignments for  $\Pi^3((q, q'))$  as follows. If  $q \neq q'$ , then accept the question  $\Pi^3((q, q'))$  by default, otherwise accept  $\Pi^3((q, q'))$  according to the verifier from the original game  $G_\ell$  on the question  $q(=q')$ .<sup>1</sup> We call such a transformation  $\mathcal{T}_3^1$  – the superscript stands for the common player's question from  $(q, q')$  and the subscript 3 stands for taking player 3's question from  $q'$  and rest of the questions from  $q$  in generating the question-triple in the new game. Succinctly, we write  $G_{\ell+1}$  as the game  $\mathcal{T}_3^1(G_\ell)$ . Likewise, we can define transformations  $\mathcal{T}_p^i$  for any  $1 \leq i, p \leq 3$ .

We show the following key properties of these transformations.

1. If  $\text{val}(G_\ell) < 1$ , then  $\text{val}(\mathcal{T}_p^i(G_\ell)) < 1$ . Furthermore,  $\mathcal{T}_p^i(G_\ell)$  remains a projection game if  $G_\ell$  is a projection game.
2. For every  $n \geq 1$ ,  $\text{val}(G_\ell^{\otimes n}) \leq \text{val}(\mathcal{T}_p^i(G_\ell)^{\otimes n})^{1/2}$ , if  $G_\ell$  is a projection game.

The first property is trivial – in the game  $\mathcal{T}_p^i(G_\ell)$ , we are still playing the game  $G_\ell$  as a subgame, and hence its value is less than 1 if  $\text{val}(G_\ell) < 1$ . For the furthermore part, we are either accepting everything by default or using the same predicate as in the original game, and hence, this transformation maintains the projection property of the game.

<sup>1</sup> Note that the way the game  $G_{\ell+1}$  is defined, the set of accepting answers for the same question-triple changes based on the underlying pair of questions  $(q, q')$ . For simplicity, we ignore this issue in this proof overview, and it will be handled in the main proof.

For the second property, we crucially use the **projection property** of the game  $G_\ell$ . We show that for any strategy  $(\alpha^1, \alpha^2, \alpha^3)$ , where  $\alpha^i : \mathcal{X}_i^n \rightarrow \mathcal{A}_i^n$ , for the game  $G_\ell^{\otimes n}$  with value  $\varepsilon$ , the *same* strategy gives value at least  $\varepsilon^2$  to the game  $\mathcal{T}_p^i(G_\ell)$ . We illustrate this for the game  $\mathcal{T}_3^1(G_\ell)$ . Let  $\mu$  be the distribution of questions from  $G_\ell$  and  $\mu|_1$  be the marginal distribution on player 1's questions, we have

$$\begin{aligned}
\varepsilon^2 &\leq \mathbf{E}_{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \sim \mu^{\otimes n}} [V((\mathbf{x}, \mathbf{y}, \mathbf{z}), (\alpha^1(\mathbf{x}), \alpha^2(\mathbf{y}), \alpha^3(\mathbf{z})))^2 \\
&= \left( \mathbf{E}_{\mathbf{v} \in \mu^{\otimes n}|_1} \left[ \mathbf{E}_{\substack{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \sim \mu^{\otimes n}, \\ \mathbf{x} = \mathbf{v}}} [V((\mathbf{x}, \mathbf{y}, \mathbf{z}), (\alpha^1(\mathbf{x}), \alpha^2(\mathbf{y}), \alpha^3(\mathbf{z}))) \right] \right)^2 \\
&\leq \mathbf{E}_{\mathbf{v} \in \mu^{\otimes n}|_1} \left[ \mathbf{E}_{\substack{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \sim \mu^{\otimes n}, \\ \mathbf{x} = \mathbf{v}}} [V((\mathbf{x}, \mathbf{y}, \mathbf{z}), (\alpha^1(\mathbf{x}), \alpha^2(\mathbf{y}), \alpha^3(\mathbf{z}))) \right]^2 \quad (\text{Cauchy-Schwarz}) \\
&= \mathbf{E}_{\mathbf{v} \in \mu^{\otimes n}|_1} \mathbf{E}_{\substack{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \sim \mu^{\otimes n}, \\ (\mathbf{x}', \mathbf{y}', \mathbf{z}') \sim \mu^{\otimes n}, \\ \mathbf{x} = \mathbf{x}' = \mathbf{v}}} \left[ \frac{V((\mathbf{x}, \mathbf{y}, \mathbf{z}), (\alpha^1(\mathbf{x}), \alpha^2(\mathbf{y}), \alpha^3(\mathbf{z}))) \cdot V((\mathbf{x}', \mathbf{y}', \mathbf{z}'), (\alpha^1(\mathbf{x}'), \alpha^2(\mathbf{y}'), \alpha^3(\mathbf{z}'))}{V((\mathbf{x}', \mathbf{y}', \mathbf{z}'), (\alpha^1(\mathbf{x}'), \alpha^2(\mathbf{y}'), \alpha^3(\mathbf{z}'))} \right]
\end{aligned}$$

Now, if we look at the triple  $(\mathbf{x}, \mathbf{y}, \mathbf{z}')$  sampled according to the above distribution, then for each  $j \in [n]$ , we have that the triple  $(x_j, y_j, z'_j)$  is distributed according to the game  $\mathcal{T}_3^1(G_\ell)$  independently. In the game,  $\mathcal{T}_3^1(G_\ell)$ , for any  $j \in [n]$  such that  $(x_j, y_j, z_j) \neq (x'_j, y'_j, z'_j)$  the new verifier is accepting by default. As for  $j \in [n]$  such that  $(x_j, y_j, z_j) = (x'_j, y'_j, z'_j)$ , the new verifier is accepting according to the original verifier on the question  $(x_j, y_j, z_j)$ . In this case, suppose  $(\alpha^1(\mathbf{x})_j, \alpha^2(\mathbf{y})_j, \alpha^3(\mathbf{z})_j)$  and  $(\alpha^1(\mathbf{x}')_j, \alpha^2(\mathbf{y}')_j, \alpha^3(\mathbf{z}')_j)$  are two satisfying assignments to the same question  $(x_j, y_j, z_j)$  according to the original game  $G_\ell$  with  $\alpha^1(\mathbf{x})_j = \alpha^1(\mathbf{x}')_j$  (as  $\mathbf{x} = \mathbf{x}'$ ), then because of the projection property of the game, we have that  $(\alpha^1(\mathbf{x})_j, \alpha^2(\mathbf{y})_j, \alpha^3(\mathbf{z}'_j))$  must be a satisfying assignment for  $(x_j, y_j, z_j)$ . As in the game  $\mathcal{T}_3^1(G_\ell)^{\otimes n}$ , we are precisely checking this for all such  $j \in [n]$ , we get that the same strategy  $(\alpha^1, \alpha^2, \alpha^3)$  gives

$$\text{val}(\mathcal{T}_3^1(G_\ell)^{\otimes n}) \geq \varepsilon^2.$$

### Putting everything together

Using the above two properties of the transformations  $\mathcal{T}_p^i$ , we conclude that if  $\text{val}(G) < 1$ , then for any  $t \geq 1$  and vectors  $\vec{i}, \vec{p} \in [3]^t$ ,

$$\text{val}(G^{\otimes n}) \leq \text{val}(\mathcal{T}_{p_t}^{i_t}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))))^{\otimes n})^{1/2^t}, \text{ and } \mathcal{T}_{p_t}^{i_t}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))) < 1.$$

Finally, we show that there exist  $t \geq 1$  and vectors  $\vec{i}, \vec{p} \in [3]^t$ , where  $t$  depends on the size of the game  $G$ , such that the game  $\mathcal{T}_{p_t}^{i_t}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G)))$  is connected (in fact, has full support). This implies that

$$\text{val}(G^{\otimes n}) \leq \text{val}(\mathcal{T}_{p_t}^{i_t}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))))^{\otimes n})^{1/2^t} \leq \exp(-\Omega_{t,G}(n)),$$

where the last inequality follows from the result of a parallel repetition theorem [10] on connected games.

## 2 Preliminaries

We start with a few notations. We use  $\mu(G)$  to denote the distribution on the questions in the game  $G$ . For  $i \in [k]$ , let  $\mu|_i$  be the marginal distribution on the questions to player  $i$ . For a  $k$ -tuple of questions  $q = (x^1, x^2, \dots, x^k)$ , we denote the question to player  $i$  by  $q|_i$ , i.e.,  $q|_1 = x^1$ ,  $q|_2 = x^2$ , and so on. For an assignment  $\alpha := (\alpha^1, \alpha^2, \dots, \alpha^k)$  to the game  $G$ , where  $\alpha^i : \mathcal{X}_i \rightarrow \mathcal{A}_i$ , and any question  $q = (x^1, x^2, \dots, x^k)$ , we use the notation  $\alpha|_q$  to denote the assignment-tuple  $(\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k))$ .

The size of the game  $G$  is referred to as the quantity  $k \cdot M \cdot \prod_{i=1}^k |\mathcal{X}_i| |\mathcal{A}_i|$ . Here, the probability of every atom in  $\text{supp}(\mu(G))$  is a multiple of  $1/M$ , where  $M$  is a finite integer. We note that as far as proving exponential decay in the  $n$ -fold repeated value of the game  $G$  with  $\text{val}(G) < 1$ , the last assumption is without loss of generality. For instance, see [18, Lemma 3.14] which lets us assume that the distribution of questions in  $G$  is uniform on the support without loss of generality.

### 2.1 Parallel repetition of connected games

As mentioned earlier, Dinur, Harsha, Venkat, and Yuen [10] showed that for a large class of  $k$ -player games, called *connected games*, the exponential decay indeed holds. Here, we define the notion of connected games for  $k$ -player games formally.

► **Definition 3** (Connected game). *A game  $G$  is called connected if for every two question pairs  $(x^1, x^2, \dots, x^k)$  and  $(x'^1, x'^2, \dots, x'^k)$  from  $\text{supp}(\mu(G))$ , there is an ordered list of questions from  $\text{supp}(\mu(G))$ ,  $((x_\ell^1, x_\ell^2, \dots, x_\ell^k))_{\ell=1}^t$  for some  $t \geq 1$ , such that the pairs  $((x^1, x^2, \dots, x^k), (x_1^1, x_1^2, \dots, x_1^k))$ ,  $((x_t^1, x_t^2, \dots, x_t^k), (x'^1, x'^2, \dots, x'^k))$ , and  $((x_\ell^1, x_\ell^2, \dots, x_\ell^k), (x_{\ell+1}^1, x_{\ell+1}^2, \dots, x_{\ell+1}^k))$  for all  $1 \leq \ell \leq t-1$  differ in only one out of the  $k$  questions.*

We will relate the value of  $G^{\otimes n}$ , where  $G$  is a projection game, with a value of  $n$ -fold parallel repetition of another game  $H$  that is connected. The following theorem shows that for connected games with a value less than 1, the value of repeated games goes down exponentially in  $n$ .

► **Theorem 4** ([10]). *For any  $k \geq 2$  and  $\varepsilon > 0$ , if  $H$  is a connected  $k$ -player game with  $\text{val}(H) = 1 - \varepsilon$ , then  $\text{val}(H^{\otimes n}) \leq \exp(-\Omega_{\varepsilon, H}(n))$ .*

### 2.2 Variants of multiplayer games

In this section, we simplify the class of games that we study. Towards this, we define the notion of *loosely-connected* games as follows.

► **Definition 5** (Loosely-connected game). *A game on the question set  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$  is loosely-connected if it is not possible to partition  $\mathcal{X}_i = \mathcal{X}'_i \cup \mathcal{X}''_i$  for all  $i \in [k]$ , so that all  $k$ -tuple of questions from the support of  $\mu(G)$  are in  $\mathcal{X}'_1 \times \mathcal{X}'_2 \times \dots \times \mathcal{X}'_k$  or  $\mathcal{X}''_1 \times \mathcal{X}''_2 \times \dots \times \mathcal{X}''_k$ .*

The following lemma states that we can assume without loss of generality that the game  $G$  is *loosely-connected*.

► **Lemma 6.** *If the exponential decay in the  $k$ -player parallel repetition holds for all projection loosely-connected games, then it also holds for all projection games.*

**Proof.** The proof of this lemma is similar to the proof of [18, Lemma 2.7]. We include the proof of this lemma here for completeness.



Let  $G$  be any projection game that is not loosely-connected with  $\mathbf{val}(G) = 1 - \varepsilon$  for some  $\varepsilon > 0$ . Without loss of generality, we can assume that there are partitions  $\mathcal{X}_i = \mathcal{X}'_i \cup \mathcal{X}''_i$  for all  $i \in [k]$  such that all the questions from the support of  $\mu(G)$  are from  $\mathcal{X}'_1 \times \mathcal{X}'_2 \times \dots \times \mathcal{X}'_k$  or  $\mathcal{X}''_1 \times \mathcal{X}''_2 \times \dots \times \mathcal{X}''_k$ , and furthermore, the game restricted to  $\mathcal{X}'_1 \times \mathcal{X}'_2 \times \dots \times \mathcal{X}'_k$  (call it  $G'$ ) and  $\mathcal{X}''_1 \times \mathcal{X}''_2 \times \dots \times \mathcal{X}''_k$  (call it  $G''$ ) are loosely-connected individually. The verifier's distribution  $\mu(G)$  on the question-tuples can be thought of as  $\mu = (1 - \delta)\mu' + \delta\mu''$  where the support of  $\mu'$  is from  $\mathcal{X}'_1 \times \mathcal{X}'_2 \times \dots \times \mathcal{X}'_k$  and the support of  $\mu''$  is from  $\mathcal{X}''_1 \times \mathcal{X}''_2 \times \dots \times \mathcal{X}''_k$ .

Now, since the value of the game  $G$  is at most  $1 - \varepsilon$ , we have  $\min\{\mathbf{val}(G'), \mathbf{val}(G'')\} \leq 1 - \varepsilon$ . Without loss of generality, suppose we have  $\mathbf{val}(G') \leq 1 - \varepsilon$ . Let the value of  $G^{\otimes n}$  be  $\eta$ . We will show that the value of the game  $G'^{\otimes n'}$  is also at least  $\eta - 2^{-\Omega_\delta(n)}$  for some  $n' = \Omega_\delta(n)$ . This will finish the proof of the lemma as we have  $\mathbf{val}(G'^{\otimes n'}) \leq \exp(-\Omega_\varepsilon(n'))$  using the fact that  $G'$  is a loosely-connected projection game.

Fix a strategy  $(\alpha^1, \alpha^2, \dots, \alpha^k)$  for  $G^{\otimes n}$  with value  $\eta$ . The  $k$ -tuple questions from  $G$  can be alternatively sampled as follows. First sample a set  $T \subseteq [n]$  by adding  $i \in T$  independently with probability  $(1 - \delta)$ . Then for each  $i \in T$ , sample a  $k$ -tuple question from the distribution  $\mu'$  independently. Similarly, for each  $i \notin T$ , sample a  $k$ -tuple question from the distribution  $\mu''$  independently. We have,

$$\mathbf{E}_{\substack{T \subseteq [n] \\ (\bar{x}^1 |_{\bar{T}}, \dots, \bar{x}^k |_{\bar{T}}) \sim \mu''^{\otimes |\bar{T}|}}} \mathbf{E}_{(\bar{x}^1 |_T, \dots, \bar{x}^k |_T) \sim \mu'^{\otimes |T|}} [V((\bar{x}^1, \bar{x}^2, \dots, \bar{x}^k), (\alpha^1(\bar{x}^1), \alpha^2(\bar{x}^2), \dots, \alpha^k(\bar{x}^k)))] = \eta.$$

From this, by the Chernoff Bound and an averaging argument, it follows that there exists  $T \subseteq [n]$  such that  $|T| \geq (1 - \delta)n/2$  and  $(\bar{y}^1, \bar{y}^2, \dots, \bar{y}^k) \sim \mu''^{\otimes |T|}$  such that

$$\mathbf{E}_{(\bar{x}^1, \bar{x}^2, \dots, \bar{x}^k) \sim \mu'^{\otimes |T|}} [V(((\bar{y}^1, \bar{x}^1), \dots, (\bar{y}^k, \bar{x}^k)), (\alpha^1((\bar{y}^1, \bar{x}^1)), \dots, \alpha^k((\bar{y}^k, \bar{x}^k))))] \geq \eta - 2^{-\Omega_\delta(n)}.$$

Here, the string  $(\bar{y}, \bar{x})$  is formed by plugging  $\bar{y}$  in the coordinates  $\bar{T}$  and  $\bar{x}$  in the coordinates  $T$ . The distribution of the questions in the expectation above precisely corresponds to the game  $G'^{\otimes |T|}$ . Thus, the strategy  $\alpha^i(\bar{y}, \bar{x}) := \alpha^i(\bar{y}^i, \bar{x}^i)$  for all  $i \in [k]$  gives the value at least  $\eta - 2^{-\Omega_\delta(n)}$  for the game  $G'^{\otimes |T|}$ .  $\blacktriangleleft$

We also consider a slight variation in the definition of  $k$ -player games, which we call *random-predicate  $k$ -player games*, where we allow a verifier to use a random predicate instead of a fixed predicate during verification.

► **Definition 7** (Random-predicate game). *A random-predicate game  $G$  is defined as follows. There exists  $R \geq 1$  such that the verifier chooses the  $k$ -tuple of questions  $(x^1, x^2, \dots, x^k)$  according to the distribution  $\mu(G)$  on the set of questions and  $r \in [R]$  uniformly at random, sends  $x^i$  to player  $i$ . The player  $i$  responds with the answer  $a^i$ . Finally, the verifier accepts the answers based on a fixed predicate  $V_r((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$ . We denote such games by  $(G, \mu, [R])$ .*

The following lemma states that for this variation of connected  $k$ -player games  $G$  the exponential decay from [10] still holds.

► **Lemma 8.** *For any connected random-predicate  $k$ -player game  $H$  and  $\varepsilon > 0$ , if  $\mathbf{val}(H) = 1 - \varepsilon$ , then  $\mathbf{val}(H^{\otimes n}) \leq \exp(-\Omega_{\varepsilon, H}(n))$ .*

**Proof.** We can think of a random-predicate  $k$ -player game  $H$  as a  $(k + 1)$ -player game  $H'$  as follows. In  $H'$ , the verifier selects the questions  $(x^1, x^2, \dots, x^k)$  from the game  $H$  and  $r \in [R]$  uniformly at random. The verifier sends  $x^i$  to players  $i$  for  $i \in [k]$ , and

sends  $r$  to player  $k + 1$ . The player  $i \in [k + 1]$  responds with the answer  $a^i$  ( $a^{k+1}$  can be anything). The verifier's predicate in  $H'$  is  $V((x^1, x^2, \dots, x^k, r), (a^1, a^2, \dots, a^k, a^{k+1})) := V_r((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$ .

It is easy to observe that if the game  $H'$  is connected then the game  $H$  is connected. Furthermore, we have  $\mathbf{val}(H^{\otimes n}) = \mathbf{val}(H'^{\otimes n})$  for any  $n \geq 1$ . Using this, the lemma follows from Theorem 4.  $\blacktriangleleft$

We can also add the projection property to a random-predicate game. The formal definition is as follows.

**► Definition 9** (Random-predicate projection game). *For any  $k \geq 2$ , a random-predicate  $k$ -player game  $(G, \mu, [R])$  is called a random-predicate projection game if for every  $k$ -tuple of question  $q = (x^1, x^2, \dots, x^k)$  and  $r \in [R]$ , there is  $D_{q,r} \geq 1$  and projections  $\sigma_{q,r}^i : \mathcal{A}_i \rightarrow [D_{q,r}]$  for  $i \in [k]$ , such that  $V_r((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$  is true iff  $\sigma_{q,r}^i(a^i) = \sigma_{q,r}^{i'}(a^{i'})$  for any  $i \neq i'$ .*

From this point onwards, we will incorporate the property of random-predicate whenever we refer to projection games.

In our proof, we will encounter random-predicate games where the choice of the verifier's predicate  $V_r$  is not uniform and may depend on the question  $q$ . The following claim says that we can assume that the distribution on verifier's predicate is uniform from a set of predicates and independent of the questions. This transformation preserves the support of the question-distribution.

**▷ Claim 10.** Suppose  $G$  is a random-predicate  $k$ -player game where on the  $k$ -tuple of question  $q \in \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$ , the distribution on the verifier's predicate  $V_r$  is sampled according to some distribution  $\nu_q$  over  $[R]$ , then there is another game  $H$  with the same distribution on the questions as in  $G$  such that the verifier for  $H$  samples a random predicate  $\tilde{V}_m$  where  $m \in [M]$  is distributed uniformly over  $[M]$ , and such that  $\mathbf{val}(G^{\otimes n}) = \mathbf{val}(H^{\otimes n})$  for all  $n \geq 1$ . Furthermore, a)  $M$  only depends on the size of the game  $G$ , and b) if  $G$  is a random-predicate projection game, then  $H$  is also a random-predicate projection game.

*Proof.* Let  $M \in \mathbb{Z}^+$  be a number such that for every question  $q$  from the game  $G$ , each atom from  $\text{supp}(\nu_q)$  has probability weight  $c/M$  for  $1 \leq c \leq M$ . In the game  $G$ , for a question  $q$  if  $\nu_q(r) = c/M$ , then in game  $H$ , we make  $c$  copies  $\tilde{V}_{i_1}(q, \cdot), \tilde{V}_{i_2}(q, \cdot), \dots, \tilde{V}_{i_c}(q, \cdot)$  of the verifier predicate  $V_r(q, \cdot)$  for the same question  $q$ . Thus, for a given question  $q$ , the verifier in  $H$  samples a random  $m \in [M]$  and decides based on the predicate  $\tilde{V}_m(q, \cdot)$ .

The a) and b) from the furthermore part follow the above construction.  $\triangleleft$

### 3 Proof of Theorem 2

Throughout this section, we fix a random-predicate  $k$ -player projection game  $(G, \mu, [R])$ , succinctly written as  $G$ , on the questions from the set  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$  and let  $\mu(G)$  be the distribution on the questions in  $G$ . Using Claim 10, we can assume without loss of generality, that along with the  $k$ -tuple questions  $(x^1, x^2, \dots, x^k) \sim \mu$ , the verifier selects  $r \in [R]$  uniformly at random, and after getting answers  $(a^1, a^2, \dots, a^k)$  from the players, applies the predicate  $V_r((x^1, x^2, \dots, x^k), (a^1, a^2, \dots, a^k))$ .

The key idea is to use the *path-trick* from [5] to relate the value of the game  $G$  (and  $G^{\otimes n}$ ) to another game  $H$  (and  $H^{\otimes n}$ ) which is connected.

### 3.1 The path-trick and the $i$ -links

In this section, we define the notion of a *link* which is analogous to the notion of the path-trick [5] that was used in connection to studying dictatorship tests towards showing hardness of approximation of constraint satisfaction problems.

Fix a player  $i \in [k]$ . An  $i$ -link from a game  $G$  is an ordered pair of original  $k$ -tuple of questions from  $G$  with possible repetition. We will induce the following distribution on  $i$ -links from  $G$ .

- Pick a question  $v$  to player  $i$  according to the distribution of  $\mu|_i$  and sample two  $k$ -tuple of questions  $q = (x^1, x^2, \dots, x^k)$  and  $q' = (y^1, y^2, \dots, y^k)$ , independently from  $\mu$  but conditioned on  $x^i = y^i = v$  and output  $(q, q')$ .

We denote the above distribution on the  $i$ -links with  $L_i(G)$ .

To see the utility of  $i$ -links, the following claim shows that the distribution  $L_i(G^{\otimes n})$  on the  $i$ -links in  $G^{\otimes n}$  is the same as the product distribution on the  $i$ -links from  $G$ .

▷ **Claim 11.** For every game  $G$ ,  $i \in [k]$ , and  $n \geq 1$ , the following two distributions are identical.

1. The distribution  $L_i(G^{\otimes n})$ .
2. The distribution  $\mathcal{D}_i$  on the  $i$ -links from  $G^{\otimes n}$  defined as follows:
  - For each  $j \in [n]$ , independently sample  $(q_j, q'_j)$  from the distribution  $L_i(G)$  where  $q_j = (x_j^1, x_j^2, \dots, x_j^k)$  and  $q'_j = (y_j^1, y_j^2, \dots, y_j^k)$ .
  - Let  $\vec{q} = (q_1, q_2, \dots, q_n)$  and  $\vec{q}' = (q'_1, q'_2, \dots, q'_n)$ . Output  $(\vec{q}, \vec{q}')$ .

Proof. Note that  $\vec{q}$  and  $\vec{q}'$ , sampled from  $\mathcal{D}_i$ , are the following  $k$ -tuple of questions

$$\begin{array}{cc} (x_1^1, x_2^1, \dots, x_n^1) & (y_1^1, y_2^1, \dots, y_n^1) \\ (x_1^2, x_2^2, \dots, x_n^2) & (y_1^2, y_2^2, \dots, y_n^2) \\ \vdots & \vdots \\ \underbrace{(x_1^k, x_2^k, \dots, x_n^k)}_{\vec{q}} & \underbrace{(y_1^k, y_2^k, \dots, y_n^k)}_{\vec{q}'} \end{array}$$

For each  $j \in [n]$ , the pair of  $k$ -tuple of questions  $(x_j^1, x_j^2, \dots, x_j^k)$  and  $(y_j^1, y_j^2, \dots, y_j^k)$  share a common pivot question  $p_j = x_j^i = y_j^i$ . This means that the question-pair  $(\vec{q}, \vec{q}')$  share a common  $n$ -tuple question  $\vec{p}$  from player  $i$ , where we think of  $\vec{q}, \vec{q}'$  as questions from the game  $G^{\otimes n}$ . This precisely corresponds to the distribution  $L_i(G^{\otimes n})$ . ◁

Consider the game  $G$  the assignments  $\alpha^i : \mathcal{X}_i \rightarrow \mathcal{A}_i$  for  $i \in [k]$ . We say that the link  $(q, q')$  from the game  $G$  is  $r$ -consistent with respect to the global assignments  $(\alpha^1, \alpha^2, \dots, \alpha^k)$  if  $q$  as well as  $q'$  are satisfied by the predicate  $V_r$  on the assignments  $(\alpha^1, \alpha^2, \dots, \alpha^k)$ .

▷ **Claim 12.** Let  $n \geq 1$ ,  $(\alpha^1, \alpha^2, \dots, \alpha^k)$  be a strategy for  $(G, \mu, [R])$  with  $\mathbf{val}(G) \geq \varepsilon$ . Then with probability at least  $\varepsilon^2$ , the link  $(q, q')$  is  $r$ -consistent with the assignments  $(\alpha^1, \alpha^2, \dots, \alpha^k)$ , where the probability is over  $(q, q')$  sampled according to  $L_i(G)$  and  $r \in [R]$  uniformly at random.

Proof. Fix the provers' strategies  $\alpha^i : \mathcal{X}_i \rightarrow \mathcal{A}_i$  for  $i \in [k]$  with value at least  $\varepsilon$ . We have,

$$\mathbf{E}_{\substack{(x^1, x^2, \dots, x^k) \sim \mu, \\ r \in [R]}} [V_r((x^1, x^2, \dots, x^k), (\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k)))] \geq \varepsilon.$$

## 54:10 Parallel Repetition of $k$ -Player Projection Games

Using the Cauchy-Schwarz inequality, we have,

$$\begin{aligned}
\varepsilon^2 &\leq \mathbf{E}_{\substack{(x^1, x^2, \dots, x^k) \sim \mu, \\ r \in [R]}} [V_r((x^1, x^2, \dots, x^k), (\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k)))]^2 \\
&= \left( \mathbf{E}_{\substack{v \in \mu|_i \\ r \in [R]}} \left[ \mathbf{E}_{\substack{(x^1, x^2, \dots, x^k) \sim \mu \\ x^i = v}} [V_r((x^1, x^2, \dots, x^k), (\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k)))] \right] \right)^2 \\
&\leq \mathbf{E}_{\substack{v \in \mu|_i \\ r \in [R]}} \left[ \mathbf{E}_{\substack{(x^1, x^2, \dots, x^k) \sim \mu \\ x^i = v}} [V_r((x^1, x^2, \dots, x^k), (\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k)))] \right]^2 \\
&\hspace{25em} \text{(Cauchy-Schwarz)} \\
&= \mathbf{E}_{\substack{v \in \mu|_i \\ r \in [R]}} \left[ \mathbf{E}_{\substack{(x^1, x^2, \dots, x^k) \sim \mu, \\ (y^1, y^2, \dots, y^k) \sim \mu, \\ x^i = y^i = v}} \left[ \begin{array}{l} V_r((x^1, x^2, \dots, x^k), (\alpha^1(x^1), \alpha^2(x^2), \dots, \alpha^k(x^k))) \\ V_r((y^1, y^2, \dots, y^k), (\alpha^1(y^1), \alpha^2(y^2), \dots, \alpha^k(y^k))) \end{array} \right] \right].
\end{aligned}$$

The expression inside the expectation above is precisely the probability that the  $i$ -link  $((x^1, x^2, \dots, x^k), (y^1, y^2, \dots, y^k))$  sampled from the distribution  $L_i(G)$  is  $r$ -consistent with respect to the assignments  $(\alpha^1, \alpha^2, \dots, \alpha^k)$ . This shows that

$$\Pr_{\substack{(q, q') \sim L_i(G) \\ r \in [R]}} [(q, q') \text{ is } r\text{-consistent with respect to the assignments } (\alpha^1, \alpha^2, \dots, \alpha^k)] \geq \varepsilon^2,$$

and this completes the proof.  $\triangleleft$

### 3.2 The transformations $\mathcal{T}_p^i$

We are now ready to define the transformation on the game  $G$  that was alluded to at the beginning of this section. We denote the transformed games by  $\mathcal{T}_p^i(G)$  for  $1 \leq i, p \leq k$ .

The distribution on the  $k$ -tuple of questions in the game  $\mathcal{T}_p^i(G)$  is over  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$  which is defined as follows.

1. The verifier samples a link  $(q, q') \sim L_i(G)$  where  $q = (x^1, \dots, x^k)$  and  $q' = (y^1, \dots, y^k)$ .
2. The verifier constructs a question-tuple by taking  $x^p$  from  $q$  and  $(y^1, \dots, y^{p-1}, y^{p+1}, \dots, y^k)$  from  $q'$ . Succinctly, we denote this operation as  $(y^1, \dots, y^{p-1}, x^p, y^{p+1}, \dots, y^k) := \Pi^p((q, q'))$  (the  $p$  stands for taking player  $p$ 's question from the first question and the remaining players' questions from the second question).

Before we define the set of satisfying assignments in the transformed game, we first define the set of  $r$ -consistent assignments, where  $r \in [R]$ , to an  $i$ -link. An assignment to a link  $(q, q')$  is an assignment to both  $q$  and  $q'$  (note that each  $k$ -tuple of question receives a separate assignment from  $\mathcal{A}_1 \times \mathcal{A}_2 \times \dots \times \mathcal{A}_k$ ). For an  $i$ -link  $(q, q')$ , we define the set of  $r$ -consistent assignments to  $(q, q')$  as follows. An  $r$ -consistent assignment to an  $i$ -link  $(q, q')$  is an assignment  $\sigma$  to the link such that

- the verifier accepts  $\sigma|_q$  on  $q$  and  $\sigma|_{q'}$  on  $q'$  according to the predicate  $V_r$ , and
- $\sigma|_{q|_i} = \sigma|_{q'|_i}$ , i.e.,  $\sigma$  gives the same value to the common question  $v$  to the player  $i$  from  $q$  and  $q'$ .

Note that an  $i$ -link  $(q, q')$  may share more than one common question with the players, but the  $r$ -consistency only cares about the question to player  $i$ .

We now define the set of accepting assignments for a  $k$ -tuple of questions in the random-predicate transformed game. The verifier chooses  $r \in [R]$  uniformly at random. Suppose a  $k$ -tuple of questions  $\tilde{q} = (z^1, z^2, \dots, z^k)$  is coming from an  $i$ -link  $(q, q')$ , i.e.,  $\tilde{q} = \Pi^p((q, q'))$ , then

- Case 1: If  $q \neq q'$ , then accept by default.
- Case 2: If  $q = q'$ , then  $\tilde{q} = q$ . In this case, accept according to the verifier's predicate  $V_r$  from the game  $G$  on question  $\tilde{q}$ .

Note that the game  $\mathcal{T}_p^i(G)$  is a random-predicate  $k$ -player game (the accepting answers for a question  $\tilde{q}$  depends on the underlying sampled link as well as the sampled  $r \in [R]$ ), and as described, the distribution on the underlying predicate is not uniform among the set of predicates. However, using Claim 10, without loss of generality, we can assume that the underlying distribution on the predicate that the verifier applies in  $\mathcal{T}_p^i(G)$  is uniform from a set of predicates (and independent of the questions). We need this as we will be applying a series of these transformations on the original game, and the transformation above is only defined on games where the verifier's predicate is uniform and independent of the question  $q$ .

In the transformed game, the verifier either accepts all the answers or accepts answers based on the predicates from the original game  $G$ . We have the following simple, but important, fact.

► **Fact 13.** *If  $G$  is a random-predicate  $k$ -player projection game, then for every  $i, p \in [k]$ , the game  $\mathcal{T}_p^i(G)$  is also a random-predicate projection game.*

### 3.2.1 Properties of the transformations $\mathcal{T}_p^i$

We start with the first claim that shows that the value of the transformed games is less than 1 if the value of  $G$  is less than 1.

▷ **Claim 14.** Fix any  $k$ -player game  $(G, \mu, [R])$ . For every  $\varepsilon \in (0, 1)$  and  $i, p \in [k]$ , if  $\text{val}(G) = 1 - \varepsilon$ , then  $\text{val}(\mathcal{T}_p^i(G)) = 1 - \varepsilon'$  where  $\varepsilon' > 0$  that depends on  $\varepsilon$  and the size of the game  $G$ .

Proof. First, observe that for every question-tuple  $q = (x^1, x^2, \dots, x^k)$  from the game  $G$ , the  $i$ -link  $(q, q)$  is present in the support of  $L_i(G)$ . Therefore, for every question-tuple  $q = (x^1, x^2, \dots, x^k)$  from the game  $\mathcal{T}_p^i(G)$  that is given by the  $i$ -link  $(q, q)$ , the verifier of the transformed game selects  $r \in [R]$  uniformly at random and uses the predicate  $V_r$ . Therefore, the transformed game  $\mathcal{T}_p^i(G)$  is a convex combination of the original game  $G$  and another game  $G'$ . Let  $\tilde{\mu}$  be the distribution on question-tuples in  $\mathcal{T}_p^i(G)$ , then it can be written as  $\tilde{\mu} = \delta\mu + (1 - \delta)\mu'$ , where  $\mu'$  corresponds to the distribution of questions from game  $G'$  and  $\delta \in (0, 1]$  that depends on the size of the game  $G$ . Thus,

$$\text{val}(\mathcal{T}_p^i(G)) \leq \delta \cdot \text{val}(G) + (1 - \delta) = \delta(1 - \varepsilon) + (1 - \delta) = 1 - \varepsilon\delta < 1. \quad \triangleleft$$

The following claim relates the value of the original game with the value of the transformed games. This claim crucially uses the fact that the original game  $G$  is a projection game.

▷ **Claim 15.** For any  $k$ -player projection game  $(G, \mu, [R])$ ,  $n \geq 1$ , and  $i, p \in [k]$ , we have  $\text{val}(\mathcal{T}_p^i(G)^{\otimes n}) \geq \text{val}(G^{\otimes n})^2$ .

## 54:12 Parallel Repetition of k-Player Projection Games

Proof. As the statement of the claim is symmetric with respect to  $p \in [k]$ , we prove the claim when  $p = 1$ . For other  $p$ , the proof is similar.

Using Claim 11, the game  $\mathcal{T}_1^i(G)^{\otimes n}$  can be described as follows: Sample an  $i$ -link  $(\vec{q}, \vec{q}')$  from the distribution  $L_i(G^{\otimes n})$ , sample  $\vec{r} \in [R]^n$  uniformly at random, and for every  $j \in [n]$  such that  $q_j = q'_j$ , apply the predicate  $V_{r_j}$  for the question  $\Pi^1((q_j, q_j))$ .

Let's fix the players' strategy  $\alpha := (\alpha^1, \alpha^2, \dots, \alpha^k)$  for the game  $G^{\otimes n}$  that gives the value  $\mathbf{val}(G^{\otimes n})$ . For an  $i$ -link  $(\vec{q}, \vec{q}')$  from  $G^{\otimes n}$ , consider the assignments  $\alpha|_{\vec{q}}$  and  $\alpha|_{\vec{q}'}$  to the link  $(\vec{q}, \vec{q}')$ . Using Claim 12, for a random  $\vec{r} \in [R]^n$  and a randomly selected  $i$ -link  $(\vec{q}, \vec{q}')$ ,  $(\vec{q}, \vec{q}')$  is  $\vec{r}$ -consistent with respect to the global assignment  $\alpha$  with probability at least  $\mathbf{val}(G^{\otimes n})^2$ . When this happens, we show that the assignment  $\alpha$  satisfies the constraint on  $\Pi^1((\vec{q}, \vec{q}'))$  from the game  $\mathcal{T}_1^i(G)^{\otimes n}$ . Indeed, pick any  $j \in [n]$  such that the  $j^{\text{th}}$  coordinate of the link  $(\vec{q}, \vec{q}')$  is  $(q_j, q_j)$  (i.e, the same question-tuple). Here, the verifier is using the predicate  $V_{r_j}$  on the question  $q_j$  in the game  $\mathcal{T}_1^i(G)^{\otimes n}$ . If the link  $(\vec{q}, \vec{q}')$ , is  $\vec{r}$ -consistent with respect to the global assignment  $\alpha$ , then we have the following

$$V_{r_j}((q_j|_1, q_j|_2, \dots, q_j|_k), (\alpha^1(\vec{q}|_1)_j, \alpha^2(\vec{q}|_2)_j, \dots, \alpha^k(\vec{q}|_k)_j)) = 1,$$

$$V_{r_j}((q_j|_1, q_j|_2, \dots, q_j|_k), (\alpha^1(\vec{q}'|_1)_j, \alpha^2(\vec{q}'|_2)_j, \dots, \alpha^k(\vec{q}'|_k)_j)) = 1.$$

In the game  $\mathcal{T}_1^i(G)^{\otimes n}$ , the verifier is checking the following condition in the coordinate  $j$ .

$$V_{r_j}((q_j|_1, q_j|_2, \dots, q_j|_k), (\alpha^1(\vec{q}|_1)_j, \alpha^2(\vec{q}'|_2)_j, \dots, \alpha^k(\vec{q}'|_k)_j)) = 1.$$

As  $G$  and the predicates  $V_r$  satisfy the projection property, and  $\vec{q}|_i = \vec{q}'|_i$  because  $(\vec{q}, \vec{q}')$  is an  $i$ -link, we see that if  $(\alpha^1(\vec{q}|_1)_j, \alpha^2(\vec{q}|_2)_j, \dots, \alpha^k(\vec{q}|_k)_j)$  and  $(\alpha^1(\vec{q}'|_1)_j, \alpha^2(\vec{q}'|_2)_j, \dots, \alpha^k(\vec{q}'|_k)_j)$  are the accepting answers for a question according to the predicate  $V_{r_j}$ , then it can be seen easily that  $(\alpha^1(\vec{q}|_1)_j, \alpha^2(\vec{q}'|_2)_j, \dots, \alpha^k(\vec{q}'|_k)_j)$  is also an accepting answer for the same question according to the same predicate  $V_{r_j}$ . This shows that the assignment  $\alpha$  passes the verifier's check on all  $j \in [n]$  such that the  $j^{\text{th}}$  coordinate of the link  $(\vec{q}, \vec{q}')$  is  $(q_j, q_j)$ . For the other coordinates, the game  $\mathcal{T}_1^i(G)$  always accepts.

Hence the same players' strategy  $\alpha$  gives  $\mathbf{val}(\mathcal{T}_1^i(G)^{\otimes n}) \geq \mathbf{val}(G^{\otimes n})^2$ .  $\triangleleft$

Finally, we compose these transformations to get a connected game, starting with a loosely-connected game. Towards this, for any string  $\beta \in ([k] \times [k])^m$ , where  $\beta_j = (\beta_j^1, \beta_j^2) \in [k] \times [k]$ , define the transformation  $\mathcal{T}^\beta(G)$  as the following transformation

$$\mathcal{T}_{\beta_2^m}^{\beta_1^1}(\dots(\mathcal{T}_{\beta_2^2}^{\beta_2^1}(\mathcal{T}_{\beta_1^1}^{\beta_1^1}(G))))).$$

For a string  $\beta$  of length  $m$ , define a string  $\beta^T$  as a  $T$  repeated copy of  $\beta$ . We have the following claim.

$\triangleright$  **Claim 16.** Let  $\beta$  be any permutation of the set  $[k] \times [k]$ . For large enough  $T \geq 1$ , the game  $\mathcal{T}^{\beta^T}(G)$  is connected (in fact, has full support) if  $G$  is loosely-connected to begin with. Furthermore,  $T \leq \prod_{i=1}^k |\mathcal{X}_i|$  which only depends on the size of the game  $G$ .

Proof. First, any transformation  $\mathcal{T}_p^i$  does not shrink the support of the questions of the previous game. By looking closely at the transformation  $\mathcal{T}_p^i$ , we conclude the following: if  $(x^1, x^2, \dots, x^k)$  and  $(y^1, y^2, \dots, y^k)$  are both in the support of  $\mu(G)$  with  $x^i = y^i$ , then the following question  $(y^1, \dots, y^{p-1}, x^p, y^{p+1}, \dots, y^k)$  will be in the support of  $\mu(\mathcal{T}_p^i(G))$ . Using this, we also observe that if we start with any game  $H$  with the property that the series of transformations  $\mathcal{T}^\beta(H)$  does not change the support of the questions, then no future transformations will change the support on the questions (as  $\beta$  contains every possible transformation  $\mathcal{T}_p^i$ ).

From the above discussion, we conclude that after some finite (only depends on the size of the game  $G$ ) series of such transformations  $\mathcal{T}^{\beta^T}(\cdot)$ , the support of the questions does not increase after another series of transformations  $\mathcal{T}^\beta$ . We denote the saturated game by  $G_{\text{final}} := \mathcal{T}^{\beta^T}(G)$  and the underlying distribution on the questions by  $\mu_{\text{final}}$ . We show that  $\mu_{\text{final}}$  has full support on  $\mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_k$ .

Suppose towards a contradiction, the support of  $\mu_{\text{final}}$  is not full. For any tuple  $q$  of length  $k$  and  $S \subseteq [k]$ , define the tuple  $q|_S$  by taking the  $S$  entries from  $q$ . Take the *smallest*  $i \in [k]$  such that there is an  $i$ -tuple  $(z^1, z^2, \dots, z^i)$  where  $(z^1, z^2, \dots, z^i) \neq q|_{[i]}$  for any  $q \in \text{supp}(\mu_{\text{final}})$ . Let  $z^* := (z^1, z^2, \dots, z^{i-1})$ . For each  $i \leq \ell \leq k$ , define the following sets.

$$\mathcal{S}_\ell = \{x \in \mathcal{X}_\ell \mid \exists q \in \text{supp}(\mu_{\text{final}}) \text{ s.t. } (z^*, x) = q|_{[i-1] \cup \{\ell\}}\}.$$

In other words,  $\mathcal{S}_\ell$  is a set of all  $x \in \mathcal{X}_\ell$  that  $(z^*, x)$  can be extended to a valid question tuple from the game  $G_{\text{final}}$ . Note that by the definition of  $z^*$ ,  $\mathcal{S}_i \subsetneq \mathcal{X}_i$  and furthermore  $\mathcal{S}_{i'} \neq \emptyset$  for any  $i' \geq i$ , as by the minimality of  $i$ , there is a valid question  $q$  in  $G_{\text{final}}$  such that  $q|_{[i-1]} = z^*$ , and hence  $q|_{i'} \in \mathcal{S}_{i'}$  for all  $i' \geq i$ .

▷ **Claim 17.** There is no  $q \in \text{supp}(\mu_{\text{final}})$  such that  $q|_i \in \overline{\mathcal{S}_i}$  and for some  $i' > i$ ,  $q|_{i'} \in \mathcal{S}_{i'}$ .

Proof. Suppose there is such a question  $q \in \text{supp}(\mu_{\text{final}})$  such that  $q|_i \in \overline{\mathcal{S}_i}$  and for some  $i' > i$ ,  $q|_{i'} \in \mathcal{S}_{i'}$ . Consider a question  $q' = (z^1, z^2, \dots, z^{i-1}, *, *, \dots)$  where  $q'|_{i'} = q|_{i'}$ . Note that such a  $q'$  is in the  $\text{supp}(\mu_{\text{final}})$  from the definition of the set  $\mathcal{S}_{i'}$ . Furthermore,  $(q, q')$  is an  $i'$ -link in the game  $G_{\text{final}}$ . Therefore, the question  $\Pi^i((q, q'))$  will be present in the game  $\mathcal{T}_{i'}^{i'}(G_{\text{final}})$ . Recall that  $\Pi^i((q, q')) = (z^1, z^2, \dots, z^{i-1}, q|_i, *, \dots)$ . However, the question  $(z^1, z^2, \dots, z^{i-1}, q|_i, *, \dots)$  is not in  $\text{supp}(\mu_{\text{final}})$  as  $q|_i \in \overline{\mathcal{S}_i}$ . This means that the game  $G_{\text{final}}$  is not saturated, which is a contradiction. ◁

This claim implies that  $\mathcal{S}_{i'} \subsetneq \mathcal{X}_{i'}$  for all  $i' > i$ . Indeed, if  $\mathcal{S}_{i'} = \mathcal{X}_{i'}$  for some  $i' > i$ , then the above claim shows that every question  $q \in \text{supp}(\mu_{\text{final}})$ ,  $q|_i \in \mathcal{S}_i$ . Hence,  $G_{\text{final}}$  (and hence  $G$ ) is not a loosely-connected game.

This claim also implies that for every question  $q \in \text{supp}(\mu_{\text{final}})$  such that  $q|_i \in \overline{\mathcal{S}_i}$ , we have  $q|_{i'} \in \overline{\mathcal{S}_{i'}}$  for every  $i' > i$ . Consider the partition of the players' question sets  $\mathcal{X}_\ell = \mathcal{X}'_\ell \cup \mathcal{X}''_\ell$  such that

- For all  $\ell \leq i-1$ ,  $\mathcal{X}'_\ell = \{z^\ell\}$ , and  $\mathcal{X}''_\ell = \mathcal{X}_\ell \setminus \mathcal{X}'_\ell$ ,
- for all  $t \geq \ell$ ,  $\mathcal{X}'_t = \mathcal{S}_\ell$ , and  $\mathcal{X}''_t = \mathcal{X}_t \setminus \mathcal{X}'_t$ .

Because  $G$  (and hence  $G_{\text{final}}$ ) is loosely connected, there must be a question  $q \in \text{supp}(\mu_{\text{final}})$  such that  $q|_{i'} \in \overline{\mathcal{S}_{i'}}$  for every  $i' \geq i$  and  $q|_t = z^t$  for some  $1 \leq t \leq i-1$ . Consider a question  $q' = (z^1, z^2, \dots, z^{i-1}, *, *, \dots)$  such that  $q'$  is in the  $\text{supp}(\mu_{\text{final}})$ . Now, the pair of questions  $(q', q)$  is an  $t$ -link in the game  $G_{\text{final}}$ , furthermore, for a questions  $q'' := \Pi^i((q, q'))$ , we have  $q'' = (z^1, z^2, z^3, \dots, z^{i-1}, q|_i, \dots)$  where  $q|_i \in \overline{\mathcal{S}_i}$ . As  $G_{\text{final}}$  is saturated,  $q'' \in \text{supp}(\mu_{\text{final}})$  but this contradicts the definition of  $\mathcal{S}_i$ . ◁

### 3.3 Finishing the proof

Let us see why these claims above are enough to prove Theorem 2.

#### Proof of Theorem 2

We start with a projection game  $G$  with  $\text{val}(G) = 1 - \varepsilon$  for some  $\varepsilon > 0$ . First, using Lemma 6, we can assume without loss of generality that  $G$  is loosely-connected. Let  $(\vec{i}, \vec{p}) \in ([k] \times [k])^{Tk^2}$  with  $(\vec{i}, \vec{p}) = \beta^T$ , i.e.,  $(i_t, p_t)$  is the  $t^{\text{th}}$  entry from the string  $\beta^T$ , where  $\beta$  and  $T$  are from Claim 16. Let  $T' = T \cdot k^2$ .

Using Claim 15 on the (random-predicate) projection game  $G$  with  $i_1, p_1 \in [k]$ , we have

$$\mathbf{val}(G^{\otimes n}) \leq \mathbf{val}(\mathcal{T}_{p_1}^{i_1}(G)^{\otimes n})^{1/2}.$$

Fact 13 shows that  $\mathcal{T}_{p_1}^{i_1}(G)$  is a projection game, and hence applying Claim 15 on the projection game  $\mathcal{T}_{p_1}^{i_1}(G)$  with  $i_2, p_2 \in [k]$ , we get

$$\mathbf{val}(\mathcal{T}_{p_1}^{i_1}(G)^{\otimes n}) \leq \mathbf{val}(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))^{\otimes n})^{1/2}.$$

Repeating this process  $T'$  times, we get

$$\mathbf{val}(G^{\otimes n}) \leq \mathbf{val}(\mathcal{T}_{p_{T'}}^{i_{T'}}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))))^{\otimes n})^{1/2^{T'}}.$$

Using Claim 14 repeatedly, we have

$$\mathbf{val}(\mathcal{T}_{p_{T'}}^{i_{T'}}(\dots(\mathcal{T}_{p_1}^{i_1}(\mathcal{T}_{p_1}^{i_1}(G)))) \leq 1 - \varepsilon',$$

where  $\varepsilon' > 0$ , that only depends on  $\varepsilon, T'$ , and the size of the game  $G$ . Finally, using Claim 16, we have that the game  $\mathcal{T}_{p_{T'}}^{i_{T'}}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))))$  is connected and hence by Lemma 8,

$$\mathbf{val}(\mathcal{T}_{p_{T'}}^{i_{T'}}(\dots(\mathcal{T}_{p_2}^{i_2}(\mathcal{T}_{p_1}^{i_1}(G))))^{\otimes n} \leq \exp(-\Omega_{\varepsilon, T, G}(n)).$$

Overall, we get  $\mathbf{val}(G^{\otimes n}) \leq \exp(-\Omega_{\varepsilon, T, G}(n))$  and the proof is completed as  $T$  only depends on the size of the original game  $G$ .

---

## References

- 1 Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *Journal of Computer and System Sciences*, 54(2):317–331, April 1997. doi:10.1006/jcss.1997.1472.
- 2 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.
- 3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, January 1998. doi:10.1145/273865.273901.
- 4 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability – Towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.
- 5 Amey Bhangale, Subhash Khot, and Dor Minzer. On Approximability of Satisfiable k-CSPs: II. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 632–642, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585120.
- 6 Amey Bhangale, Swastik Kopparty, and Sushant Sachdeva. Simultaneous approximation of constraint satisfaction problems. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan*, volume 9134, pages 193–205, 2015. doi:10.1007/978-3-662-47672-7\_16.
- 7 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 335–340, New York, NY, USA, 2015. ACM. doi:10.1145/2746539.2746565.
- 8 Mark Braverman, Subhash Khot, and Dor Minzer. Parallel repetition for the ghz game: Exponential decay. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1337–1341, 2023. doi:10.1109/FOCS57990.2023.00080.
- 9 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcg and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005. doi:10.1137/S0097539704443057.



- 10 Irit Dinur, Prahladh Harsha, Rakesh Venkat, and Henry Yuen. Multiplayer Parallel Repetition for Expanding Games. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67, pages 37:1–37:16, Dagstuhl, Germany, 2017. doi:10.4230/LIPIcs.ITCS.2017.37.
- 11 Irit Dinur, Oded Regev, and Clifford Smyth. The hardness of 3-uniform hypergraph coloring. *Combinatorica*, 25(5):519–535, 2005. doi:10.1007/s00493-005-0032-4.
- 12 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014. doi:10.1145/2591796.2591884.
- 13 Uriel Feige. A threshold of  $\ln n$  for approximating set cover. *J. ACM*, 45(4):634–652, July 1998. doi:10.1145/285055.285059.
- 14 Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996. doi:10.1145/226643.226652.
- 15 H. Furstenberg and Y. Katznelson. A density version of the hales-jewett theorem for  $k=3$ . *Discrete Mathematics*, 75(1):227–241, 1989. doi:10.1016/0012-365X(89)90089-7.
- 16 Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel Repetition for the GHZ Game: A Simpler Proof. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207, pages 62:1–62:19, Dagstuhl, Germany, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.62.
- 17 Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for all 3-player games over binary alphabet. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 998–1009, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520071.
- 18 Uma Girish, Kunal Mittal, Ran Raz, and Wei Zhan. Polynomial Bounds on Parallel Repetition for All 3-Player Games with Binary Inputs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245, pages 6:1–6:17, Dagstuhl, Germany, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.6.
- 19 Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. Going beyond bell’s theorem. In Menas Kafatos, editor, *Bell’s Theorem, Quantum Theory and Conceptions of the Universe*, pages 69–72. Springer Netherlands, Dordrecht, 1989. doi:10.1007/978-94-017-0849-4\_10.
- 20 Venkatesan Guruswami, Johan Hastad, and Madhu Sudan. Hardness of approximate hypergraph coloring. *SIAM Journal on Computing*, 31(6):1663–1686, 2002. doi:10.1137/S0097539700377165.
- 21 Thomas Holenstein. Parallel Repetition: Simplification and the No-Signaling Case. *Theory of Computing*, 5(1):141–172, 2009. doi:10.4086/toc.2009.v005a008.
- 22 Justin Holmgren and Ran Raz. A parallel repetition theorem for the GHZ game. *arXiv preprint arXiv:2008.05059*, 2020.
- 23 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001. doi:10.1145/502090.502098.
- 24 S. Khot. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 23–32, 2002. doi:10.1109/SFCS.2002.1181879.
- 25 Subhash Khot. Hardness results for approximate hypergraph coloring. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, STOC ’02*, pages 351–359, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.509962.
- 26 Kunal Mittal and Ran Raz. Block Rigidity: Strong Multiplayer Parallel Repetition Implies Super-Linear Lower Bounds for Turing Machines. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185, pages 71:1–71:15, Dagstuhl, Germany, 2021. doi:10.4230/LIPIcs.ITCS.2021.71.
- 27 DHJ Polymath. A new proof of the density Hales-Jewett theorem. *Annals of Mathematics*, pages 1283–1327, 2012. doi:10.4007/annals.2012.175.3.6.

## 54:16 Parallel Repetition of k-Player Projection Games

- 28 Anup Rao. Parallel Repetition in Projection Games and a Concentration Bound. *SIAM Journal on Computing*, 40(6):1871–1891, 2011. doi:10.1137/080734042.
- 29 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.
- 30 Oleg Verbitsky. Towards the Parallel Repetition Conjecture. *Theor. Comput. Sci.*, 157(2):277–282, May 1996. doi:10.1016/0304-3975(95)00165-4.

# Faster Algorithms for Schatten- $p$ Low Rank Approximation

Praneeth Kacham   

Carnegie Mellon University, Pittsburgh, PA, USA  
Google Research, New York, USA

David P. Woodruff   

Carnegie Mellon University, Pittsburgh, PA, USA

---

## Abstract

We study algorithms for the Schatten- $p$  Low Rank Approximation (LRA) problem. First, we show that by using fast rectangular matrix multiplication algorithms and different block sizes, we can improve the running time of the algorithms in the recent work of Bakshi, Clarkson and Woodruff (STOC 2022). We then show that by carefully combining our new algorithm with the algorithm of Li and Woodruff (ICML 2020), we can obtain even faster algorithms for Schatten- $p$  LRA.

While the block-based algorithms are fast in the real number model, we do not have a stability analysis which shows that the algorithms work when implemented on a machine with polylogarithmic bits of precision. We show that the LazySVD algorithm of Allen-Zhu and Li (NeurIPS 2016) can be implemented on a floating point machine with only logarithmic, in the input parameters, bits of precision. As far as we are aware, this is the first stability analysis of any algorithm using  $O((k/\sqrt{\varepsilon}) \text{poly}(\log n))$  matrix-vector products with the matrix  $A$  to output a  $1 + \varepsilon$  approximate solution for the rank- $k$  Schatten- $p$  LRA problem.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Mathematical optimization; Mathematics of computing  $\rightarrow$  Mathematical analysis

**Keywords and phrases** Low Rank Approximation, Schatten Norm, Rectangular Matrix Multiplication, Stability Analysis

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.55

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.11959>

**Acknowledgements** We would like to thank Cameron Musco, Christopher Musco, and Aleksandros Sobczyk for helpful discussions. We thank a Simons Investigator Award and NSF CCF-2335411 for partial support.

## 1 Introduction

Low Rank Approximation (LRA) is an important primitive in large scale data analysis. Given an  $m \times n$  matrix  $A$ , and a rank parameter  $k$ , the task is to find a rank- $k$  matrix  $B$  that minimizes  $\|A - B\|$  where  $\|\cdot\|$  is some matrix norm. Typically, we also require that the algorithms output a factorization  $B = XY$  such that  $X \in \mathbb{R}^{m \times k}$  and  $Y \in \mathbb{R}^{k \times n}$ . Such a factorization lets us compute the product  $Bz$  with an arbitrary vector  $z$  in time  $O(k(n + m))$  which can be significantly smaller than the  $\text{nnz}(A)$  time required to multiply a vector with the original matrix  $A$ . Here  $\text{nnz}(A)$  denotes the number of non-zero entries of the matrix  $A$ . Thus, replacing  $A$  with a low rank approximation can make downstream tasks much faster. Additionally, if the matrix  $A$  has a low rank structure but is corrupted by noise, a low rank approximation of  $A$  can recover the underlying structure under suitable assumptions on the noise. We note that many low rank approximation algorithms, including ours, compute a rank- $k$  orthonormal matrix  $W$  such that  $\|A(I - WW^\top)\|$  is small and then define  $X = AW$  and  $Y = W^\top$ .



© Praneeth Kacham and David P. Woodruff;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 55; pp. 55:1–55:19



Leibniz International Proceedings in Informatics

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

In this paper, the error metric we consider is given by the Schatten- $p$  norm for  $p \geq 1$ . Given a matrix  $M$ , the Schatten- $p$  norm of  $M$  denoted by  $\|M\|_{S_p}$  is defined as  $(\sum_i \sigma_i(M)^p)^{1/p}$  where  $\sigma_i(M)$  denotes the  $i$ -th singular value of  $M$ . Note that Schatten-2 norm is the same as the Frobenius norm, denoted by  $\|M\|_F = (\sum_{i,j} M_{ij}^2)^{1/2}$  and the Schatten- $\infty$  norm is the same as the operator norm, denoted by  $\|M\|_2 = \max_{x \neq 0} \|Mx\|_2 / \|x\|_2$ . In the presence of outliers, the Schatten-1 norm,  $\sum_i \sigma_i(M)$ , is considered to be more robust since the errors introduced by the outliers are not “squared” as it is done in the case of the Frobenius norm.

The Schatten- $p$  norm low rank approximation problem asks to find a rank- $k$  matrix  $B$  that minimizes  $\|A - B\|_{S_p}$ . As the Schatten- $p$  norms are unitarily invariant, we have from Eckart-Young-Mirsky’s theorem that  $\|A - A_k\|_{S_p} = \min_{\text{rank-}k B} \|A - B\|_{S_p}$  for all  $p \geq 1$ , where  $A_k$  is the matrix obtained by truncating the Singular Value Decomposition (SVD) of  $A$  to only the top  $k$  singular values. This implies that a single matrix  $A_k$  is a *best* rank- $k$  approximation for  $A$  for all values of  $p$ . However, computing the SVD of an  $m \times n$  matrix takes  $O(\min(mn^{\omega-1}, nm^{\omega-1}))$  time (see Appendix A), where  $\omega$  is the matrix multiplication exponent. This time complexity is prohibitive when  $m$  and  $n$  are large. Thus, we relax the requirements and ask for a rank- $k$  matrix  $B$  satisfying  $\|A - B\|_{S_p} \leq (1 + \varepsilon)\|A - A_k\|_{S_p}$  in the hope of obtaining faster algorithms than the SVD.

While a single matrix  $A_k$  is a best low rank approximation for  $A$  in all Schatten- $p$  norms, it is not the case for approximate solutions, i.e., if  $B$  is a rank- $k$  matrix that satisfies  $\|A - B\|_{S_p} \leq (1 + \varepsilon)\|A - A_k\|_{S_p}$  for some  $p$ , it may not be the case that  $\|A - B\|_{S_q} \leq (1 + \varepsilon)\|A - A_k\|_{S_q}$  for  $q \neq p$ . Thus, many approximation algorithms for Schatten- $p$  LRA are tailored to the particular value  $p$ . There are two different lines of works for Schatten- $p$  LRA in the literature: (i) Sketching based algorithms of Li and Woodruff [9] and (ii) Iterative algorithms of Bakshi, Clarkson and Woodruff [2]. We summarize the running times of the algorithms in Table 1. The sketch-based algorithms are usually non-adaptive and the iterative algorithms adaptively pick their matrix-vector product queries depending on the results in the previous round which makes them powerful as we can see from the superior running time over sketch-based algorithms when we desire solutions with small  $\varepsilon$ .

**Sketching Algorithms.** Li and Woodruff [9] gave (almost) input-sparsity time algorithms for Schatten- $p$  LRA, extending the earlier input-sparsity time algorithms for Frobenius norm LRA from [4]. For  $p < 2$ , their algorithm runs in  $\tilde{O}(\text{nnz}(A) + \max(m, n) \cdot \text{poly}(k/\varepsilon))$  time and for  $p > 2$ , their algorithm runs in  $\tilde{O}(\text{nnz}(A) + \max(m, n) \cdot \min(m, n)^{\alpha_p} \text{poly}(k/\varepsilon))$  time, where  $\alpha_p = (\omega - 1)(1 - 2/p)$ . Note that for the current value of  $\omega \approx 2.37$ , their algorithm runs in  $\Omega(mn)$  time for  $p \geq 7.4$  and hence is not an “input-sparsity time” algorithm but for all constant  $p, k, \varepsilon$ , their algorithm runs in  $o(\min(mn^{\omega-1}, nm^{\omega-1}))$  time and therefore is faster than computing the SVD.

■ **Table 1** Running times for  $1 + \varepsilon$  rank- $k$  Schatten- $p$  LRA algorithms for  $m \times n$  matrices assuming  $m \geq n$ .

|  | Time Complexity   |
|--|---|
| Li and Woodruff [9] ( $p \in [1, 2)$ ) | $O(\text{nnz}(A) \log n) + \tilde{O}_p(mk^{2(\omega-1)/p} / \varepsilon^{(4/p-1)(\omega-1)})$<br>$+ \tilde{O}_p(k^{2\omega/p} / \varepsilon^{(4/p-1)(2\omega+2)})$                |
| Li and Woodruff [9] ( $p > 2$ )        | $O(\text{nnz}(A) \log n) + \tilde{O}_p(n^{\omega(1-2/p)} k^{2\omega/p} / \varepsilon^{2\omega/(p+2)})$<br>$+ \tilde{O}_p(mn^{(\omega-1)(1-2/p)} (k/\varepsilon)^{2(\omega-1)/p})$ |
| Bakshi et al. [2]                      | $O(p^{1/6} \varepsilon^{-1/3} \text{nnz}(A) k \log(n/\varepsilon) + mp^{(\omega-1)/6} k^{\omega-1} \varepsilon^{-(\omega-1)/3})$  |

**Iterative Algorithms.** Recently, Bakshi, Clarkson, and Woodruff [2] gave an iterative algorithm for Schatten- $p$  LRA. Their algorithm runs the Block Krylov iteration algorithm of Musco and Musco [11] at *two* different block sizes for different number of iterations respectively. They show that the algorithm succeeds in computing a low rank approximation at one of the block sizes and show how to compute which block size succeeds in computing the approximation. For Schatten- $p$  LRA, their algorithm requires  $O(kp^{1/6} \text{poly}(\log n)/\varepsilon^{1/3})$  matrix-vector products with the matrix  $A$  and hence can be implemented in  $\tilde{O}(\text{nnz}(A)kp^{1/6}/\varepsilon^{1/3})$  time. At a high level, their algorithm runs the Block Krylov iteration algorithm with block size  $k$  for  $O(p^{1/6}\varepsilon^{-1/3} \text{poly}(\log n))$  iterations and with block size  $O(p^{-1/3}\varepsilon^{-1/3}k)$  for  $O(\sqrt{p} \text{poly}(\log n))$  iterations. They set these parameters such that the algorithm requires an overall same number of matrix-vector products with  $A$  at both block sizes. They argue that for a matrix with a “flat” spectrum, the low rank approximation computed by the block size  $k$  algorithm is a  $1 + \varepsilon$  approximation and for a matrix with a “non-flat” spectrum, the solution computed by block size  $O(p^{-1/3}\varepsilon^{-1/3}k)$  algorithm is a  $1 + \varepsilon$  approximation.

**Comparison.** As we can see from Table 1, the running times of these algorithms depend in a quite complicated way on the parameters  $\text{nnz}(A)$ ,  $m$ ,  $n$ ,  $\varepsilon$  and  $p$ . Throughout the paper, we assume that  $m = n$ ,  $\text{nnz}(A) = n^2$  (i.e., the matrix  $A$  is dense) and  $k \leq n^c$  for a small constant  $c$  so that  $k \ll n$ . In some cases, where sparsity in the datasets cannot be well exploited, such as when processing the datasets using GPUs, it is natural to analyze the time complexities of the algorithms and compare the performances assuming that the inputs are dense.

For  $p \in [1, 2)$ , we have that the time complexity of the algorithm of [9] is  $O(n^2 \log n + n \text{poly}(k)/\varepsilon^{(4/p-1)(\omega-1)} + \text{poly}(k)/\varepsilon^{(4/p-1)(2\omega+2)})$  and the time complexity of the algorithm of [2] is  $O(\varepsilon^{-1/3}n^2k \log(n) + n \text{poly}(k)/\varepsilon^{(\omega-1)/3})$ . We see that only when

$$1/\varepsilon > n^{\frac{1}{(4/p-1)(\omega+1)-1/6}},$$

the algorithm of [2] is faster than the sketching based algorithm of [9]. For  $\omega \approx 2.371$  and  $p = 1$ , the above is achieved only when  $1/\varepsilon \geq n^{\approx 0.1}$ . Hence, in the high accuracy regime, the algorithm of [2] is faster than that of the sketching based algorithm of [9]. For other values of  $p \in [1, 2)$ ,  $\varepsilon$  has to be even smaller than  $1/n^{0.1}$  for the algorithm of [2] to be faster than the algorithm of [9].

For comparing the algorithms in the case  $p > 2$ , first we pick  $\varepsilon$  to be a constant and obtain that the running time of the algorithm of [9] is  $O(n^2 \log n + n^{1+(\omega-1)(1-2/p)} \text{poly}(k))$  and the algorithm of [2] has a running time of  $O(p^{1/6}n^2k \log(n))$ . Thus, as long as  $(\omega-1)(1-2/p) \leq 1$ , the sketch-based algorithm is faster than the iterative algorithm. We call  $p$  such that  $(\omega-1)(1-2/p) \leq 1$ , the *crossover point* from “sketch” to “iterative”. For the current value of  $\omega \approx 2.371$ , the crossover point is  $\approx 7.39$ .

Now consider the case of  $\varepsilon = 1/n$  and constant  $p$ . The iterative algorithm of [2] has a running time of  $O(n^{2+1/3}k \log(n))$  and the sketch based algorithm of [9] has a running time of  $O(n^\omega \text{poly}(k))$  and thus offers no improvement over the naïve SVD algorithm. This again shows that in the high precision regime, the small dependence on  $\varepsilon$  in the running time of the algorithm of [2] is crucial to obtain better than  $O(n^\omega)$  time algorithm. Overall, we summarize the comparison between the algorithms in Table 2.

**Our Improvements.** We first *improve* the time complexity of the iterative algorithm of [2] for *all* parameter regimes. While the focus of their paper was to minimize the number of matrix-vector products required, we observe that by using fast rectangular matrix multiplication algorithms, we can obtain even faster algorithms using their technique of running the block

■ **Table 2** In the case of  $m = n$ ,  $\text{nnz}(A) = n^2$  and  $k = n^{o(1)}$ , the table lists which of the previous works is asymptotically faster for the current value of  $\omega \approx 2.371$ . **Iterative** algorithm refers to the algorithm of [2] and the **Sketching** algorithm refers to the algorithm of [9]. In the above, crossover  $\approx 7.4$ .

|                            | Small $\varepsilon$ ( $\approx 1/n$ ) | Large $\varepsilon$ |
|----------------------------|---------------------------------------|---------------------|
| $p \in [1, 2)$             | Iterative                             | Sketching           |
| $2 < p < \text{crossover}$ | Iterative                             | Sketching           |
| $p > \text{crossover}$     | Iterative                             | Iterative           |

Krylov iteration algorithm at different block sizes. Fast rectangular matrix multiplication algorithms let us obtain a different block-size vs iteration trade-off giving us faster algorithms. This algorithm directly achieves the fastest running times for small  $\varepsilon$  since we improve upon [2] in all regimes.

We saw above that for constant  $\varepsilon$ , the sketch based algorithm takes only  $O(n^2 \log n)$  time when  $p \lesssim 7.4$  and hence cannot be improved upon over asymptotically by more than  $\text{polylog}(n)$  factors in that regime. We show that using a combination of our fast iterative algorithm and the algorithm of [9] gives an algorithm that runs in near-linear time<sup>1</sup> for all  $p \lesssim 22$  for appropriate  $\varepsilon$  values extending the values of  $p$  for which a Schatten- $p$  LRA can be computed in  $O(n^2 \log n)$  time, when the rank parameter  $k \leq n^c$ .

Our combined algorithm works as follows: to solve a sub-problem in the algorithm of [9], we run our improved iterative algorithm for Schatten- $p$  LRA with accuracy parameter  $\varepsilon = 1/n$ . As our improved iterative algorithm has a better dependence on  $\varepsilon$  than earlier algorithms, we obtain a faster algorithm for solving the sub-problem and hence obtain an  $O(n^2 \log n)$  time algorithm for all  $p \lesssim 22$ . Thus, improving the performance of iterative algorithms in the small  $\varepsilon$  regime let us obtain faster algorithms overall in the large  $\varepsilon$  regime!

### Numerically Stable Algorithms

While the algorithm of [2] and our modification give fast algorithms for Schatten- $p$  Low Rank Approximation, it is not known if the Block Krylov iteration algorithm is stable when implemented on a floating point machine with  $O(\log(n/\varepsilon))$  bits of precision. It is a major open question in numerical linear algebra to show if the Block Krylov iteration algorithm is stable. Obtaining fast algorithms that provably work on finite precision machines is a tricky problem in general. We note that until the recent work of Banks, Garza-Vargas, Kulkarni and Srivastava [3], it was not clear if an eigendecomposition of a matrix could be computed in  $\tilde{O}(n^\omega)$  time on a finite precision machine. Building on these ideas, another recent work [14] obtains fast and stable algorithms for the generalized eigenvalue problem. The sketch-and-solve methods, such as the algorithm of [9], are usually stable as the operations do not blow up the magnitude of the entries. As we note above, for large  $p$ , the algorithms in [9] are not input-sparsity time and hence an important question is if there are any stable input-sparsity time algorithms for large  $p$ . We answer this question in affirmative by showing that the LazySVD algorithm of [1] can be stably implemented on a floating point machine with  $O(\log m\kappa/\varepsilon)$  bits of precision where  $\kappa = \sigma_1(A)/\sigma_{k+1}(A)$ . The LazySVD algorithm computes a low rank approximation for all  $p \geq 2$ .

<sup>1</sup> Note the near-linear here means  $\tilde{O}(n^2)$  as the input-matrix is assumed to have  $n^2$  nonzero entries.

Similar to the Block Krylov iteration algorithm, LazySVD also needs  $O(k \text{ poly}(\log n)/\sqrt{\varepsilon})$  matrix-vector products with  $A$ . Additionally, the factorization output by LazySVD is *simultaneously* a  $1 + \varepsilon$  approximation for all  $p \geq 2$ . To find a rank- $k$  approximation of  $A$ , the LazySVD algorithm first computes a unit vector  $v$  which is an approximation to the top eigenvector of  $A^\top A$ . Then the algorithm deflates  $A^\top A$  and forms the matrix  $(I - vv^\top)A^\top A(I - vv^\top)$  and proceeds to find an approximation to the top eigenvector of  $(I - vv^\top)A^\top A(I - vv^\top)$  and so on for a total of  $k$  rounds. The authors show that the span of  $k$  vectors found across all the iterations contains a  $1 + \varepsilon$  approximation if the eigenvector approximations satisfy an appropriate condition. Thus, to implement the LazySVD algorithm on a floating point machine, we first need a stable routine that can compute approximations to the top eigenvector of a given matrix. We show that such a routine can be implemented stably using the Lanczos algorithm [12]. We additionally modify the LazySVD algorithm and show that the modification allows us to compute matrix-vector products with the deflated matrix to a good enough approximation which lets the Lanczos algorithm compute an approximation to the top eigenvector of the deflated matrix. Our slight modification to LazySVD turns out to be important in making the stability analysis go through.

The novelty of our stability analysis is that instead of showing each of the vectors  $\tilde{v}_1, \dots, \tilde{v}_k$  computed by a finite precision algorithm are close to the vectors  $v_1, \dots, v_k$  that would be computed by an algorithm with unbounded precision, we essentially argue that for all  $i$ , the projection matrices onto the subspaces spanned by  $\tilde{v}_1, \dots, \tilde{v}_i$  and  $v_1, \dots, v_i$  are close using induction. This change makes the stability analysis work with only a polylogarithmic number of bits of precision whereas showing all  $\tilde{v}_i$ s are individually close to corresponding  $v_i$ s would require polynomially many bits of precision.

### 1.1 Our Results

In the following,  $\alpha$  denotes the constant such that an arbitrary  $n \times n$  matrix can be multiplied with an arbitrary  $n \times n^\alpha$  matrix using  $O(n^{2+\eta})$  arithmetic operations for any constant  $\eta > 0$ . The matrix multiplication exponent  $\omega$  is the smallest constant such that an arbitrary  $n \times n$  matrix can be multiplied with an arbitrary  $n \times n$  matrix using  $O(n^{\omega+\eta})$  arithmetic operations for any constant  $\eta > 0$ . For simplicity, we ignore the constant  $\eta$ , and write as if the matrices can be multiplied in  $O(n^\omega)$  time. We define  $\beta := (\omega - 2)/(1 - \alpha)$ . Note that  $\beta \leq 1$ .<sup>2</sup>

► **Theorem 1** (Informal, Theorem 5). *Given an  $n \times n$  matrix  $A$ , a rank parameter  $k$  and an accuracy parameter  $\varepsilon$ , there is an algorithm that outputs a rank- $k$  orthonormal matrix  $W$  that with probability  $\geq 0.9$  satisfies,  $\|A(I - WW^\top)\|_{S_p} \leq (1 + O(\varepsilon))\|A - A_k\|_{S_p}$ . If  $k \leq \varepsilon \cdot n^\alpha$ , then the algorithm runs in  $\tilde{O}(\sqrt{pn}^{2+\eta})$  time for any constant  $\eta > 0$ .*

Combining the algorithm in the above theorem and the algorithm of [9], we obtain the following result:

► **Theorem 2** (Informal, Theorem 8). *Given an  $n \times n$  matrix  $A$ , a rank parameter  $k$  independent of  $n$  and any constant  $\eta > 0$ , there is a randomized algorithm that runs in time  $\tilde{O}((n^{1-2/p})^{2+\eta+(1-\alpha)\beta/(1+2\beta)} \text{ poly}(1/\varepsilon) + n^2)$  and outputs a rank- $k$  projection  $\hat{Q}$  that satisfies  $\|A(I - \hat{Q})\|_{S_p}^p \leq (1 + \varepsilon)\|A - A_k\|_{S_p}^p$ , with probability  $\geq 0.9$*

The above theorem shows that for all  $p$  at most a suitable constant, the algorithm runs in  $\tilde{O}(n^2)$  time for  $\varepsilon > 1/n^{c_p}$  for a small enough constant  $c_p$  and hence is faster than using the algorithm of [9] or the algorithm in Theorem 1.

<sup>2</sup> See Section 2.2.

The following result shows that our modification of LazySVD can be stably implemented on a floating point machine.

► **Theorem 3 (Informal, Theorem 11).** *Given an  $n \times d$  matrix  $A$  with condition number  $\kappa(A) = \sigma_1(A)/\sigma_{k+1}(A)$ , an accuracy parameter  $\varepsilon$ , a rank parameter  $k$  and probability parameter  $\eta$ , if the machine precision  $\varepsilon_{\text{mach}} \leq \text{poly}(\varepsilon\eta/n\kappa(A))$ , then there is an algorithm that outputs a  $d \times k$  matrix  $V_k$  such that  $\kappa(V_k) \leq 4$  and with probability  $\geq 1 - \eta$ , for all  $p \in [2, \infty]$ ,*

$$\|A(I - \text{Proj}_{\text{colspace}(V_k)})\|_{S_p} \leq (1 + O(\varepsilon))\|A - A_k\|_{S_p},$$

and runs in time  $O(\frac{\text{nnz}(A)k}{\sqrt{\varepsilon}} \text{poly}(\log(d\kappa(A)/\varepsilon\eta)) + d \text{poly}(k, \log(dk/\eta\varepsilon)))$ .

In the above theorem,  $\text{Proj}_{\text{colspace}(M)}$  denotes the orthogonal projection matrix onto the column space of  $M$ .

## 1.2 Implications to Practice

While the theoretical fast rectangular matrix multiplication algorithms are not practically efficient, the message of this paper is that by optimizing for the number of matrix-vector products as in [2], we are leaving a lot of performance on the table. In modern computing architectures, multiplying an  $n \times n$  and an  $n \times b$  matrix is, for example, much faster than  $b$  times the time required to multiply the  $n \times n$  and an  $n \times 1$  vector because of data locality and the opportunities for parallelization. Thus, in the algorithm of [2], running the block size  $k$  version for fewer iterations while increasing the larger block size  $b$  can give faster algorithms in practice than using the parameters that optimize for the number of matrix-vector products. We include a small experiment in the appendix which compares the time required to compute the product of an  $n \times n$  matrix with matrices that have different numbers of columns.

LazySVD with our stability analysis uses a similar number of matrix vector products as the widely used Block Krylov iteration algorithm while requiring only polylogarithmic bits of precision. While as mentioned above, block-based algorithms such as Block Krylov iteration can be much faster than single-vector algorithms such as LazySVD and our modification of it, it is only the case when the matrix is directly given to us. When the matrix is implicitly defined in other ways (for e.g., as the Hessian of a neural network where we can efficiently compute Jacobian-Vector products), the difference in performance between block-based algorithms and single-vector algorithms is less pronounced. When guarantees of stability are required, the fastest algorithms in practice for Low Rank Approximation should use some combination of sketching as in [9] to reduce dimension stably and then use our modification of LazySVD algorithm to find the necessary top  $k$  subspace.

## 2 Preliminaries

### 2.1 Notation

For a positive integer  $n$ , we use  $[n]$  to denote the set  $\{1, \dots, n\}$ . We use the notation  $\tilde{O}(f(n))$  to denote  $O(f(n) \text{poly}(\log(f(n))))$  and  $\tilde{O}_q(f(n))$  to hide the multiplicative factors that depend only on the parameter  $q$ . For a vector  $x$ , we use  $\|x\|_2 = (\sum_i |x_i|^2)^{1/2}$  to denote the Euclidean norm of  $x$ . Given an  $m \times n$  matrix  $A$ , we use  $A_{i,j}$  to denote the entry in the index  $(i, j)$  of  $A$ . We use  $A_{i*}$  to denote the  $i$ -th row of  $A$  and  $A_{*j}$  to denote the  $j$ -th column. We identify the multiplication of an  $m \times n$  matrix with an  $n \times k$  matrix with the notation  $[m, n, k]$ . For a matrix  $A$ , we use  $\text{colspace}(A)$  to denote the vector space  $\{Ax \mid x \in \mathbb{R}^n\}$ . For



any vector space  $V \in \mathbb{R}^n$ , we use  $\text{Proj}_V$  to denote the linear operator which maps a vector  $x$  to the projection of  $x$  in the subspace  $V$  i.e., the nearest vector to  $x$  in  $V$  in terms of Euclidean distance. If the columns of  $X$  are an orthonormal basis for  $V$ , then  $\text{Proj}_V = XX^\top$ .

Let  $A = U\Sigma V^\top$  be the singular value decomposition (SVD) of  $A$  and let  $\sigma_1 \geq \dots \geq \sigma_n$  (recall  $m \geq n$ ) denote the singular values of  $A$ . For  $k \leq n$ , let  $A_k := \sum_{i=1}^k \sigma_i U_{*i} (V^\top)_{i*}$  be the matrix obtained by truncating the SVD of  $A$  to the top  $k$  singular values.

We use  $\|A\|_F$  to denote the Frobenius norm  $(\sum_{i,j} A_{i,j}^2)^{1/2}$  and  $\|A\|_2$  to denote the operator norm  $\max_{x \neq 0} \|Ax\|_2 / \|x\|_2$ . For  $p \geq 1$ , we define  $\|A\|_{S_p} = (\sum_{i=1}^n \sigma_i^p)^{1/p}$  to be the Schatten- $p$  norm. As  $\|\cdot\|_{S_p}$  defines a norm, we have  $\|A+B\|_{S_p} \leq \|A\|_{S_p} + \|B\|_{S_p}$  for any two  $m \times n$  matrices  $A$  and  $B$ . Additionally, we have  $\|A^\top\|_{S_p} = \|A\|_{S_p}$  and for any unitary matrices  $U', V'$ , we have  $\|U'AV'\|_{S_p} = \|A\|_{S_p}$ .

## 2.2 Fast Rectangular Matrix Multiplication

Let  $\omega$  denote the best matrix multiplication exponent. The current upper bound on  $\omega$  is  $\approx 2.371$  [5] and for  $\gamma < 1$ , let  $\omega(\gamma)$  denote the exponent such that the product of an  $n \times n$  with an  $n \times n^\gamma$  matrix can be computed using  $O(n^{\omega(\gamma)+n})$  arithmetic operations for any constant  $\eta > 0$ . There exists  $\alpha > 0.31$  [8, 6] such that for all  $\gamma < \alpha$ ,  $\omega(\gamma) = 2$  and for all  $\gamma \geq \alpha$ ,

$$\omega(\gamma) \leq 2 + (\omega - 2) \frac{\gamma - \alpha}{1 - \alpha}.$$

See [7, 10] for the above bound on  $\omega(\gamma)$ . Recall  $\beta := \frac{\omega-2}{1-\alpha}$ . We now observe that  $n^{1-\alpha}n^2 \geq n^\omega$  since a matrix product of the form  $[n, n, n]$  can be computed using  $n^{1-\alpha}$  matrix products of the form  $[n, n, n^\alpha]$ . Hence,  $1 - \alpha \geq \omega - 2$ , which implies  $\beta \leq 1$ .

## 3 Schatten- $p$ LRA using Fast Matrix Multiplication

■ **Algorithm 1** Block Krylov Iteration Algorithm [11].

---

**Input:** An  $n \times n$  matrix  $A$ , rank parameter  $k$ , block size  $b$  and number  $q$  of iterations

**Output:** An orthonormal matrix  $Z \in \mathbb{R}^{n \times k}$

- 1  $\Pi \sim \mathcal{N}(0, 1)^{n \times b}$
  - 2  $K \leftarrow [A\Pi \quad (AA^\top)A\Pi \quad \dots \quad (AA^\top)^q A\Pi]$  // The Krylov Matrix
  - 3 Orthonormalize columns of  $K$  to get an  $n \times qb$  matrix  $Q$
  - 4 Compute  $M := Q^\top AA^\top Q$
  - 5 Set  $\bar{U}_k$  to the top  $k$  singular vectors of  $M$
  - 6 **return**  $Z = Q\bar{U}_k$
- 

### 3.1 Block Krylov Iteration Algorithm

The block Krylov Iteration algorithm of Musco and Musco [11] is stated as Algorithm 1. For any  $b$ , let  $T(n, b)$  be the time to multiply an  $n \times n$  matrix with an  $n \times b$  matrix. The Block Krylov iteration algorithm with rank parameter  $k$ , block size  $b \geq k$  and iteration count  $q$  (with  $bq \leq n$ ) runs in time at most  $(2q+1)T(n, b) + n(qb)^{\omega-1} + 3T(n, qb) + (qb)^\omega + T(n, k)$ .<sup>3</sup>

---

<sup>3</sup> Assuming that SVD of the  $qb \times qb$  matrix  $M$  in Algorithm 1 can be computed in time  $O((qb)^\omega)$ .

Using the fact that  $T(n, qb) \leq qT(n, b)$  and  $qb \leq n$ , we obtain that the time complexity of the algorithm is  $O(qT(n, b) + n(qb)^{\omega-1})$ . We now have  $T(n, b) \geq (b/n)n^\omega$  since otherwise the matrix product of the form  $[n, n, n]$  can be computed quicker than in  $n^\omega$  time by computing the  $n/b$  products of the form  $[n, n, b]$ . Hence,  $qT(n, b) \geq qbn^{\omega-1} \geq n(qb)^{\omega-1}$  using  $qb \leq n$ . Thus, we obtain that the time complexity of the Block Krylov Iteration algorithm with parameters  $k, b, q$  satisfying  $b \geq k$  and  $bq \leq n$  is  $O(qT(n, b))$ . We now state a few properties of the Block Krylov algorithm that we use throughout the paper.

► **Theorem 4.** *With a large probability over the Gaussian matrix  $\Pi$ , the following properties hold for the matrix  $Z$  computed by Algorithm 1:*

1. *There is a universal constant  $c$  such that for all  $i \in [k]$ ,*

$$\sigma_i(Z^\top A)^2 \geq \|A^\top(Z)_{*i}\|_2^2 \geq \sigma_i^2 - (c \log^2 n/q^2)\sigma_{k+1}^2.$$

*This follows from the per-vector error guarantee of Theorem 1 in [11].*

2. *If  $\text{gap} := (\sigma_k/\sigma_{b+1}) - 1$  and  $q \geq C \log(n/\varepsilon)/\sqrt{\min(1, \text{gap})}$  for a large enough constant  $C$ , then for all  $i \in [k]$ ,  $\sigma_i(Z^\top A)^2 \geq \|A^\top(Z)_{*i}\|_2^2 \geq \sigma_i^2 - \varepsilon\sigma_{k+1}^2$ .*

The second guarantee in the above theorem follows from the gap-dependent error bounds in Theorem 11 in [11]. Note the logarithmic dependence of  $q$  on  $1/\varepsilon$ .

■ **Algorithm 2** Schatten- $p$  Norm Subspace Approximation.

---

**Input:** An  $n \times n$  matrix  $A$ , rank parameter  $k$  and an accuracy parameter  $\varepsilon$

**Output:** Approximate Solution to the Schatten- $p$  Norm Subspace Approximation problem

$$1 \quad q \leftarrow \begin{cases} \sqrt{p} & k \leq \varepsilon \cdot n^\alpha \\ \max(\sqrt{p}, p^{\frac{1}{2(1+2\beta)}} (k/n^\alpha \varepsilon)^{\frac{\beta}{1+2\beta}}) & \varepsilon \cdot n^\alpha \leq k \leq n^\alpha \\ \max(\sqrt{p}, p^{\frac{1}{2(1+2\beta)}} / \varepsilon^{\frac{\beta}{1+2\beta}}) & k \geq n^\alpha \end{cases}$$

$$2 \quad b' \leftarrow \lceil (3/2) \max(1, k/q^2 \varepsilon) \rceil$$

$$3 \quad Z_1 \leftarrow \text{BLOCKKRYLOV}(A, \text{rank} = k, \text{block size} = k, \text{iterations} = O(q \log(n)))$$

$$4 \quad Z_2 \leftarrow \text{BLOCKKRYLOV}(A, \text{rank} = k, \text{block size} = b' + k, \text{iterations} = O(\sqrt{p} \log(n/\varepsilon)))$$

$$5 \quad W_1 \leftarrow \text{colspan}(A^\top Z_1)$$

$$6 \quad W_2 \leftarrow \text{colspan}(A^\top Z_2)$$

$$7 \quad W \leftarrow W_2 \text{ if } \hat{\sigma}_k \geq (1 + 1/2p)\hat{\sigma}_{b'+k} \text{ and } W_1 \text{ otherwise // These approximations to } \sigma_k \text{ and } \sigma_{b'+k} \text{ can be computed using the } M \text{ matrix computed in}$$

Algorithm 1

---

### 3.2 Main Theorem

► **Theorem 5.** *Given an  $n \times n$  matrix  $A$ , a rank parameter  $k$  and an accuracy parameter  $\varepsilon$ , Algorithm 2 outputs a  $k$  dimensional orthonormal matrix  $W$  that with probability  $\geq 0.9$  satisfies,  $\|A(I - WW^\top)\|_{S_p} \leq (1 + O(\varepsilon))\|A - A_k\|_{S_p}$ . For any constant  $\eta > 0$ , the running time of the algorithm is as follows:*

1. *For  $k \leq \varepsilon n^\alpha$ , the algorithm runs in time  $\tilde{O}(\sqrt{p}n^{2+\eta})$ .*

2. *For  $\varepsilon n^\alpha \leq k \leq n^\alpha$ , the algorithm runs in time  $\tilde{O}(\max(\sqrt{p}n^{2+\eta}, p^{\frac{1}{2(1+2\beta)}} n^{2+\eta} (k/n^\alpha \varepsilon)^{\beta/(1+2\beta)}))$ .*

3. *For  $k \geq n^\alpha$ , the algorithm runs in time  $\tilde{O}((p^{1/2} \varepsilon^{-\beta})^{1/(1+2\beta)} n^{2+\eta-\alpha\beta} k^\beta)$ .*

Assuming  $p$  is a constant independent of  $\varepsilon$ , the dependence on  $\varepsilon$  is at least better than  $\varepsilon^{-1/3}$  as  $\beta \leq 1$  which implies  $\beta/(1+2\beta) \leq 1/3$ . The proof of this theorem is similar to that of [2].

We include the proof in the full version.

#### 4 Comparison with the Algorithm of Li and Woodruff [9]

For  $n \times n$  matrices and  $p > 2$ , the algorithm of [9] for the Schatten- $p$  norm Subspace Approximation problem, shown in Algorithm 3 runs in time

$$O(n^2 \log n) + \tilde{O}_p \left( \frac{n^{\omega(1-2/p)} k^{2\omega/p}}{\varepsilon^{2\omega/p+2}} + n^{1+(\omega-1)(1-2/p)} (k/\varepsilon)^{2(\omega-1)/p} \right). \quad (1)$$

Let  $K = k + \varepsilon/\eta_1 = k + n^{1-2/p} k^{2/p} / \varepsilon^{2/p}$ . To obtain the above running time, they use a ridge leverage score sampling algorithm to compute a matrix  $S$  with  $s = O(\varepsilon^{-2} K \log n)$  rows that satisfies (2) with a large probability. The same guarantee can instead be obtained by using the Sub-sampled Randomized Hadamard Transform (SRHT) [16] with  $s = O(\varepsilon^{-2} K \log n)$  rows and the matrix-product  $SA$  can be computed in time  $O(n^2 \log n)$ . To obtain the subspace embedding guarantee for  $T$  as required in Algorithm 3, we can let the matrix  $T$  again be an SRHT with  $r = O(\varepsilon^{-2} s \log n)$  columns and the product  $SAT$  can be computed in time  $O(ns \log s) = O(\varepsilon^{-2} n(k + n^{1-2/p} k^{2/p} / \varepsilon^{2/p}))$ .

The singular value decomposition of the matrix  $SAT$  can be computed in  $O(rs^{\omega-1}) = O(\varepsilon^{-2\omega} (k + n^{1-2/p} k^{2/p} / \varepsilon^{2/p})^\omega \text{polylog}(n))$  time and a basis for the rowspace of  $W^\top SA$  can be computed in  $O(skn)$  time. Overall, for constant  $k$  and  $\varepsilon$ , the algorithm of [9] runs in time  $\tilde{O}(n^2 + (n^{1-2/p})^\omega)$ . For  $p > 2\omega/(\omega - 2)$ , their algorithm runs in  $n^{2+c_p}$  time for a constant  $c_p > 0$  that depends on  $p$ . For the same parameters, our algorithm runs in  $\tilde{O}(n^2)$  time and hence we have an improvement. For  $k \leq n^\alpha$  and  $\varepsilon = 1/n$ , their algorithm runs in time  $\Omega(n^\omega)$  which means that computing the SVD of  $A$  is already faster whereas our algorithm runs in time  $\tilde{O}(n^{2+\frac{(1-\alpha)\beta}{1+2\beta}}) = o(n^\omega)$  if  $\omega > 2$ . Hence, our algorithm improves upon the algorithm of [9] for a wide range of parameters. We note that computing the SVD of  $SAT$  turns out to be the most expensive step for large  $p$ . In the next subsection, we show that our Algorithm 2 can be used to sidestep the computation of the SVD of  $SAT$ , thereby giving an even faster algorithm.

We call  $p^* = 2\omega/(\omega - 2)$ , the *crossover* point. For  $p > p^*$ , our Algorithm 2 is faster than the algorithm of [9]. For the current value of  $\omega \approx 2.37$ ,  $p^* \approx 12.8$ . For  $p < p^*$ , the leading order term in the time complexity of Algorithm 3 is  $O(n^2 \log n)$  for  $\varepsilon > n^{-c_p}$  for a constant  $c_p$  depending on  $p$ , and hence is faster than Algorithm 2.

■ **Algorithm 3** Schatten- $p$  Norm Low Rank Approximation for  $p > 2$  [9].

---

**Input:** A matrix  $A \in \mathbb{R}^{m \times n}$  and an accuracy parameter  $\varepsilon$

**Output:** A rank- $k$  orthonormal projection  $Q$  satisfying

$$\|A(I - Q)\|_{S_p} \leq (1 + \varepsilon) \|A - A_k\|_{S_p}$$

1  $\eta_1 \leftarrow O(\varepsilon^{1+2/p} / k^{2/p} n^{1-2/p})$

2  $S$  be a matrix with  $s$  rows that satisfies

$$(1 - \varepsilon) A^\top A - \eta_1 \|A - A_k\|_F^2 \cdot I \preceq A^\top S^\top SA \preceq (1 + \varepsilon) A^\top A + \eta_1 \|A - A_k\|_F^2 \cdot I. \quad (2)$$

3  $T \leftarrow$  Subspace embedding for  $s$ -dimensional subspaces with error  $O(\varepsilon)$

4  $W \leftarrow$  Top  $k$  left singular vectors of  $SAT$

5  $Z \leftarrow$  Matrix whose columns are an orthonormal basis for the row space of  $W^\top SA$

6  $Q \leftarrow ZZ^\top$

---

#### 4.1 Further Improving the running time of [9] using our algorithm

Given an  $n \times n$  matrix  $A$ ,  $p \geq 1$  and  $r \leq n$ , let  $\|A\|_{(p,r)} = (\sum_{i=1}^r \sigma_i(A)^p)^{1/p}$ . We can show that  $\|\cdot\|_{(p,r)}$  is a norm over  $n \times n$  matrices. As  $\|\cdot\|_{(p,r)}$  is unitarily invariant, we have by Eckart-Young-Mirsky's theorem that  $\|A - A_k\|_{(p,r)} = \min_{\text{rank-}k B} \|A - B\|_{(p,r)}$ . In Lemma 4.2 of [9], they show that for  $S$  satisfying (2), if  $\hat{Q}$  is a rank- $k$  projection matrix with

$$\|SA(I - \hat{Q})\|_{(p,r)} \leq (1 + \varepsilon) \min_{\substack{\text{rank-}k \\ \text{projections } Q}} \|SA(I - Q)\|_{(p,r)}, \quad (3)$$

then  $\|A(I - \hat{Q})\|_{S_p}^p \leq (1 + C_p \varepsilon) \|A - A_k\|_{S_p}^p$ , for a constant  $C_p$  that only depends on  $p$ . They show that the matrix  $Q$  returned by Algorithm 3 satisfies (3) and then conclude that the matrix  $Q$  is a  $1 + O(\varepsilon)$  approximation to the Schatten- $p$  norm low rank approximation problem. We will now argue that there is a faster algorithm for computing a projection that satisfies (3). The algorithm does not require the computation of the SVD of the matrix  $SA$  and hence does not incur the  $O_{p,k,\varepsilon}(n^{(1-2/p)\omega})$  term in the running time. We first show that a  $1 + \varepsilon$  approximate solution to the Schatten- $p$  norm subspace approximation problem, is a  $1 + \varepsilon n/r$  approximation to the  $(p, r)$  subspace approximation problem.

► **Lemma 6.** *For an arbitrary  $m \times n$  matrix  $A$  ( $m \leq n$ ), if  $\hat{Q}$  is a rank- $k$  projection matrix satisfying  $\|A(I - \hat{Q})\|_{S_p}^p \leq (1 + \varepsilon) \|A - A_k\|_{S_p}^p$  and  $\text{colspan}(\hat{Q}) \subseteq \text{rowspan}(A)$ , then for any  $r \leq n$ ,*

$$\|A(I - \hat{Q})\|_{(p,r)}^p \leq (1 + \varepsilon \lceil (m - k)/r \rceil) \|A - A_k\|_{(p,r)}^p.$$

**Proof.** Let  $\hat{Q}$  be a rank- $k$  projection such that

$$\|A(I - \hat{Q})\|_{S_p}^p \leq (1 + \varepsilon) \min_{\text{rank-}k \text{ projections } Q} \|A(I - Q)\|_{S_p}^p = (1 + \varepsilon) \sum_{i=k+1}^n \sigma_i(A)^p.$$

Note that  $\|A(I - \hat{Q})\|_{S_p}^p = \sum_{i=1}^{m-k} \sigma_i(A(I - \hat{Q}))^p$  since the matrix  $A(I - \hat{Q})$  has rank at most  $m - k$  from our assumption that  $\text{colspan}(\hat{Q}) \subseteq \text{rowspan}(A)$ . Now,  $\|A(I - \hat{Q})\|_{(p,r)}^p = \sum_{i=1}^r \sigma_i(A(I - \hat{Q}))^p$  and therefore,

$$\begin{aligned} \|A(I - \hat{Q})\|_{(p,r)}^p &= \|A(I - \hat{Q})\|_{S_p}^p - \sum_{i=r+1}^{m-k} \sigma_i(A(I - \hat{Q}))^p \\ &\leq (1 + \varepsilon) \sum_{i=k+1}^m \sigma_i(A)^p - \sum_{i=r+1}^{m-k} \sigma_i(A(I - \hat{Q}))^p. \end{aligned}$$

Since the matrix  $A\hat{Q}$  has rank at most  $k$ , by Weyl's inequality,  $\sigma_i(A(I - \hat{Q})) \geq \sigma_{i+k}(A)$  which implies

$$\begin{aligned} \|A(I - \hat{Q})\|_{(p,r)}^p &\leq \sum_{i=k+1}^{k+r} \sigma_i(A)^p + \varepsilon \|A - A_k\|_{S_p}^p + \left( \sum_{i=k+r+1}^m \sigma_i(A)^p - \sum_{i=r+1}^{m-k} \sigma_i(A(I - \hat{Q}))^p \right) \\ &\leq \min_{\text{rank-}k \text{ projections } Q} \|A(I - Q)\|_{(p,r)}^p + \varepsilon \|A - A_k\|_{S_p}^p. \end{aligned}$$

Finally, using the fact that  $\|A - A_k\|_{S_p}^p \leq \lceil (m - k)/r \rceil \|A - A_k\|_{(p,r)}^p$ , we obtain

$$\|A(I - \hat{Q})\|_{(p,r)}^p \leq (1 + \varepsilon \lceil (m - k)/r \rceil) \|A - A_k\|_{(p,r)}^p. \quad \blacktriangleleft$$

Finally, we have the following lemma which shows how to find an approximate solution to the  $(p, r)$  Low Rank Approximation problem.

► **Lemma 7.** *Let  $A \in \mathbb{R}^{m \times n}$  be an arbitrary matrix with  $m \leq n$ . Given parameters  $k, p, r$  and  $\varepsilon$ , there is a randomized algorithm to find a rank- $k$  projection  $\hat{Q}$ , that with probability  $\geq 9/10$  satisfies,*

$$\|A(I - \hat{Q})\|_{(p,r)}^p \leq (1 + \varepsilon) \|A - A_k\|_{(p,r)}^p.$$

For constant  $p$  and  $k \leq m^\alpha$  and any constant  $\eta > 0$ , the randomized algorithm runs in time  $\tilde{O}(m^{2+\eta+(1-\alpha)\beta/(1+2\beta)} k^{\beta/(1+2\beta)} \text{poly}(1/\varepsilon) + nm + nk^{\omega-1})$  and for  $k \geq m^\alpha$ , the algorithm runs in  $\tilde{O}(m^{2+\eta-\alpha\beta+\frac{\beta}{1+2\beta}} k^{\beta} \text{poly}(1/\varepsilon) + nm^{1-\alpha\beta} k^{\beta} + nk^{\omega-1})$  time.

**Proof.** First we note that

$$\min_{\text{rank-}k \text{ projections } Q} \|A(I - Q)\|_{(p,r)}^p = \min_{\text{rank-}k \text{ projections } W} \|(I - W)A\|_{(p,r)}^p = \|A - A_k\|_{(p,r)}^p.$$

Let  $T$  be an SRHT matrix with  $O(\varepsilon^{-2} m \text{polylog}(n))$  rows. With a large probability,  $T$  is an  $\varepsilon$  subspace embedding for the rowspace of matrix  $A$ . Then

$$(1 - \varepsilon)AA^\top \preceq ATT^\top A^\top \preceq (1 + \varepsilon)AA^\top$$

and further for all rank- $k$  projections  $W$ ,

$$(1 - \varepsilon)(I - W)AA^\top(I - W) \preceq (I - W)ATT^\top A^\top(I - W) \preceq (1 + \varepsilon)(I - W)AA^\top(I - W).$$

We then have for all  $i$  that  $\sigma_i((I - W)AT) = (\sqrt{1 \pm \varepsilon})\sigma_i((I - W)A)$ . Therefore,  $\|(I - W)AT\|_{(p,r)}^p = (1 \pm \varepsilon)^{p/2} \|(I - W)A\|_{(p,r)}^p$  for all rank- $k$  projections  $W$ . Let Algorithm 2 be run on the matrix  $T^\top A^\top$  with rank parameter  $k$  and approximation parameter  $\varepsilon/pm$ . By Theorem 5, we obtain a rank- $k$  projection  $\widehat{W}$  satisfying

$$\|T^\top A^\top(I - \widehat{W})\|_p \leq (1 + \varepsilon/pm) \min_{\text{rank-}k \text{ projections } W} \|T^\top A^\top(I - W)\|_p$$

Using, Lemma 6, we obtain that

$$\|T^\top A^\top(I - \widehat{W})\|_{(p,r)}^p \leq (1 + \varepsilon) \min_{\text{rank-}k \text{ projections } W} \|T^\top A^\top(I - W)\|_{(p,r)}^p.$$

By using the relation between  $\|(I - W)AT\|_{(p,r)}^p$  and  $\|(I - W)A\|_{(p,r)}^p$  for all projections  $W$ , we get

$$\begin{aligned} \|A^\top(I - \widehat{W})\|_{(p,r)}^p &\leq \frac{(1 + \varepsilon)^{p/2+1}}{(1 - \varepsilon)^{p/2}} \min_{\text{rank-}k \text{ projections } W} \|A^\top(I - W)\|_{(p,r)}^p \\ &\leq (1 + O(\varepsilon p)) \|A - A_k\|_{(p,r)}^p. \end{aligned}$$

Now,  $\|A - A(\widehat{W}A)^\dagger(\widehat{W}A)\|_{(p,r)}^p \leq \|A - \widehat{W}A\|_{(p,r)}^p = \|(I - \widehat{W})A\|_{(p,r)}^p \leq (1 + O(\varepsilon p)) \|A - A_k\|_{(p,r)}^p$ . Scaling  $\varepsilon$ , we obtain the result.

**Runtime Analysis.** The matrix  $AT$  can be computed in time  $O(mn \log n)$ . For constant  $p$ , Algorithm 2 runs on the matrix  $T^\top A$  in time  $\tilde{O}(m^{2+\eta+(1-\alpha)\beta/(1+2\beta)} k^{\beta/(1+2\beta)} \text{poly}(1/\varepsilon))$  for  $k \leq m^\alpha$  and in time  $\tilde{O}(m^{2+\eta-\alpha\beta+\frac{\beta}{1+2\beta}} k^{\beta} \text{poly}(1/\varepsilon))$  for  $k \geq m^\alpha$ . Finally, the rowspace of  $\widehat{W}^\top A$  can be computed in time  $O(nm + nk^{\omega-1})$  for  $k \leq m^\alpha$  and  $O(nm^{1-\alpha\beta} k^{\beta} + nk^{\omega-1})$  for  $k \geq m^\alpha$ . ◀

Using the above lemma, we can find a rank- $k$  projection  $\hat{Q}$  that satisfies

$$\|SA(I - \hat{Q})\|_{(p,r)}^p \leq (1 + \varepsilon)\|A - A_k\|_{(p,r)}^p$$

in time  $\tilde{O}((n^{1-2/p})^{2+\eta+(1-\alpha)\beta/(1+2\beta)} \text{poly}(1/\varepsilon) + n^2)$  for constant  $k$  improving on the  $\tilde{O}(n^2 + (n^{1-2/p})^\omega \text{poly}(1/\varepsilon))$  running time of [9] for the current value of  $\omega$  since  $2 + \frac{(1-\alpha)\beta}{1+2\beta} = 2 + \frac{\omega-2}{1+2\beta} < \omega$  if  $\beta \neq 0$ . We thus have the following theorem.

► **Theorem 8.** *Given a dense  $n \times n$  matrix  $A$ , a constant rank parameter  $k$  and any constant  $\eta > 0$ , there is a randomized algorithm that runs in time  $\tilde{O}((n^{1-2/p})^{2+\eta+(1-\alpha)\beta/(1+2\beta)} \text{poly}(1/\varepsilon) + n^2)$  and outputs a rank- $k$  projection  $\hat{Q}$  that, with probability  $\geq 9/10$ , satisfies  $\|A(I - \hat{Q})\|_{S_p}^p \leq (1 + \varepsilon)\|A - A_k\|_{S_p}^p$ .*

For this algorithm, the crossover point is  $\tilde{p} = \frac{4(1+2\beta)}{\omega-2} + 2$  i.e., only when  $p > \tilde{p}$ , Algorithm 2 is faster than the algorithm in the above theorem for constant  $k$  and  $\varepsilon$ . For current values of  $\omega, \alpha$ , we have  $\tilde{p} \approx 22$ . In particular, for constant  $k$  and  $\varepsilon > n^{-c_p}$ , for  $p \lesssim 22$ , the algorithm has a time complexity of only  $\tilde{O}(n^2)$ .

## 5 Stability of LazySVD

### 5.1 Finite Precision Preliminaries

Following the presentation of [12], we say that a floating point machine has precision  $\varepsilon_{\text{mach}}$  if it can perform computations to relative error  $\varepsilon_{\text{mach}}$ . More formally, let  $\text{fl}(x \circ y)$  be the result of the computation  $x \circ y$  on the floating point machine where  $\circ \in \{+, -, \times, \div\}$ . We say that the floating point machine has a precision  $\varepsilon_{\text{mach}}$  if for all  $x$  and  $y$ ,  $\text{fl}(x \circ y) = (1 + \delta)(x \circ y)$  where  $|\delta| \leq \varepsilon_{\text{mach}}$ . Additionally, we also require  $\text{fl}(\sqrt{x}) = (1 + \delta)\sqrt{x}$  for some  $\delta$  with  $|\delta| \leq \varepsilon_{\text{mach}}$ . Ignoring overflow or underflow, a machine which implements the IEEE floating point standard with  $\geq \log_2(1/\varepsilon_{\text{mach}})$  bits of precision satisfies the above requirements (see [12, Section 5]). Given matrices  $A$  and  $B$  with at most  $n$  rows and columns, we can compute a matrix  $C$ , on a floating point machine, that satisfies  $\|C - A \cdot B\|_2 \leq \varepsilon_{\text{mach}} \text{poly}(n)\|A\|_2\|B\|_2$  by directly computing  $C_{ij}$  as  $\text{fl}(\sum_k A_{ik}B_{kj})$ .

### 5.2 Stability Analysis

■ **Algorithm 4** LazySVD [1].

---

**Input:** A positive semidefinite matrix  $M \in \mathbb{R}^{d \times d}$ ,  $k \leq d, \varepsilon, \varepsilon_{\text{pca}}, \eta$   
**Output:** Vectors  $v_1, \dots, v_k$

- 1  $M_0 \leftarrow M$  and  $V_0 \leftarrow []$
- 2 **for**  $s = 1, \dots, k$  **do**
- 3      $v'_s \leftarrow \text{AppxPCA}_{\varepsilon/2, \varepsilon_{\text{pca}}, \eta/k}(M_{s-1})$
- 4      $v_s \leftarrow (I - V_{s-1}V_{s-1}^\top)v'_s / \|(I - V_{s-1}V_{s-1}^\top)v'_s\|_2$
- 5      $V_s \leftarrow [V_{s-1} \ v_s]$
- 6      $M_s \leftarrow (I - V_sV_s^\top)M(I - V_sV_s^\top)$  // The matrix  $M_s$  is not computed as we only need matrix vector products with  $M_s$
- 7 **return**  $V_k$

---

The LazySVD algorithm (Algorithm 4) of [1] crucially requires a routine called **AppxPCA** that computes an approximation to the top eigen vector of the given positive semidefinite matrix. While they use a particular **AppxPCA** algorithm in their results, any routine that satisfies the following definition can be plugged into the LazySVD algorithm.

► **Definition 9** (AppxPCA). We say that an algorithm is **AppxPCA** with parameters  $\varepsilon, \varepsilon_{\text{pca}}$  and  $\eta$  if given a positive semidefinite matrix  $M \in \mathbb{R}^{d \times d}$  with an orthonormal set of eigenvectors  $u_1, \dots, u_d$  corresponding to eigenvalues  $\lambda_1 \geq \dots \geq \lambda_d \geq 0$ , the algorithm outputs a unit vector  $w$  such that with probability  $\geq 1 - \eta$ ,  $\sum_{i \in [d]: \lambda_i \leq (1-\varepsilon)\lambda_1} \langle w, u_i \rangle^2 \leq \varepsilon_{\text{pca}}$ .

We now show that Lanczos algorithm can be used to stably compute a vector that satisfies the AppxPCA guarantee.

► **Lemma 10.** *If for any vector  $x$ , we can compute a vector  $y$  such that*

$$\|y - M_s x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(n)\kappa) \|M_s\|_2 \|x\|_2$$

and if  $\varepsilon_{\text{mach}} \leq \text{poly}(\varepsilon_{\text{pca}}\eta/n\kappa)$ , then we can compute a unit vector  $v$  such that with probability  $\geq 1 - \eta$ ,  $\sum_{i: \lambda_i(M_s) \leq (1-\varepsilon)\lambda_1(M_s)} \langle v, u_i(M_s) \rangle^2 \leq \varepsilon$ . The algorithm uses  $O(\frac{1}{\sqrt{\varepsilon}} \text{poly}(\log(d/\varepsilon\eta\varepsilon_{\text{pca}})))$  matrix vector products with  $M_s$ .

**Proof.** Let  $\mathbf{z}$  be a  $d$  dimensional random vector with each coordinate being an independent Gaussian random variable. Let  $M_s = \sum_i \lambda_i u_i u_i^\top$  be the eigendecomposition. Let  $r$  be the largest index such that  $\lambda_r \geq (1 - \varepsilon)\lambda_1$ . Consider the vector  $M_s^q \mathbf{z}$  for a  $q$  we choose later. We have

$$\mathbf{y} = M_s^q \mathbf{z} = \sum_{i=1}^d \lambda_i^q \langle u_i, \mathbf{z} \rangle u_i.$$

Consider  $\langle u_1, \mathbf{z} \rangle$ . By 2-stability of Gaussian random variables,  $\langle u_1, \mathbf{z} \rangle \sim N(0, \|u_1\|_2^2) = N(0, 1)$ . Hence with probability  $1 - \eta$ ,  $|\langle u_1, \mathbf{z} \rangle| \geq \eta$ . We also have that with probability  $\geq 1 - \eta$ , for all  $i = 1, \dots, d$   $|\langle u_i, \mathbf{z} \rangle| \leq O(\sqrt{\log d/\eta})$ . Condition on these events. Now,  $\|\mathbf{y}\|_2^2 = \sum_{i=1}^d \lambda_i^{2q} \langle u_i, \mathbf{z} \rangle^2 \geq \lambda_1^{2q} \langle u_1, \mathbf{z} \rangle^2 \geq \lambda_1^{2q} \eta^2$ . Define  $\hat{\mathbf{y}} = \mathbf{y}/\|\mathbf{y}\|_2$ . Let  $i > r$  so that  $\lambda_i < (1 - \varepsilon)\lambda_1$  by definition of  $r$ . We have

$$|\langle u_i, \hat{\mathbf{y}} \rangle| = \frac{|\langle u_i, \mathbf{y} \rangle|}{\|\mathbf{y}\|_2} \leq \frac{\lambda_i^q |\langle u_i, \mathbf{z} \rangle|}{\lambda_1^q \eta} \leq \frac{\lambda_i^q \sqrt{\log d/\eta}}{\lambda_1^q \eta} \leq (1 - \varepsilon)^q \frac{C \sqrt{\log d/\eta}}{\eta}.$$

If  $q \geq C\varepsilon^{-1} \log(d/\varepsilon_{\text{pca}}\eta)$  for a large enough constant  $C$ , we get  $|\langle u_i, \hat{\mathbf{y}} \rangle| \leq \text{poly}(\varepsilon_{\text{pca}}/d)$ . Thus,  $\sum_{i=r+1}^d |\langle u_i, \hat{\mathbf{y}} \rangle|^2 \leq \text{poly}(\varepsilon_{\text{pca}})$ .

Now define  $f(x) = x^q$  so that  $f(M_s)\mathbf{z} = \mathbf{y}$  and define  $\rho = \lambda_1/q$ . From [13, Chapter 3] there is a polynomial  $p(x)$  of degree  $\sqrt{2q \log 1/\gamma}$  such that for all  $x \in [-\rho, \lambda_1 + \rho]$ ,

$$|p(x) - x^q| \leq e\gamma\lambda_1^q.$$

As we can compute matrix-vector products with  $M_s$  up to an additive error of  $O(\varepsilon_{\text{mach}} \text{poly}(n)\kappa)$ , using Theorem 1 of [12] as long as  $\varepsilon_{\text{mach}} \leq \varepsilon'\rho/(\text{poly}(n)\kappa\|M_s\|_2) \leq \varepsilon'/\text{poly}(n)\kappa$ , we can compute a vector  $\mathbf{y}'$  on a floating point machine, using  $\sqrt{2q \log 1/\gamma}$  iterations such that

$$\begin{aligned} \|\mathbf{y} - \mathbf{y}'\|_2 &= \|(M_s)^q \mathbf{z} - \mathbf{y}'\|_2 \leq ((7e\gamma\sqrt{2q \log 1/\gamma})\lambda_1^q + \varepsilon'\lambda_1^q) \|\mathbf{z}\|_2 \\ &\leq O(\gamma\sqrt{2q \log 1/\gamma} + \varepsilon')\lambda_1^q \sqrt{d}. \end{aligned}$$

where we used that  $\|\mathbf{z}\|_2 \leq O(\sqrt{d})$  with high probability. As  $\|\mathbf{y}\|_2 \geq \lambda_1^q \eta$ , we further obtain that

$$\|\mathbf{y} - \mathbf{y}'\|_2 \leq O(\gamma\sqrt{2q \log 1/\gamma} + \varepsilon')\sqrt{d}\|\mathbf{y}\|_2/\eta.$$

## 55:14 Schatten- $p$ Low Rank Approximation

We set  $\gamma = \text{poly}(\varepsilon_{\text{pca}}\eta/dq)$  and  $\varepsilon' = \text{poly}(\varepsilon_{\text{pca}}\eta/d)$  to obtain that  $\|\mathbf{y} - \mathbf{y}'\|_2 \leq \text{poly}(\varepsilon_{\text{pca}}/d)\|\mathbf{y}\|_2$ . Thus,

$$\|\hat{\mathbf{y}} - \mathbf{y}'/\|\mathbf{y}'\|_2\|_2 \leq \|\mathbf{y}/\|\mathbf{y}\|_2 - \mathbf{y}'/\|\mathbf{y}'\|_2\|_2 \leq \text{poly}(\varepsilon_{\text{pca}}/d).$$

On a floating point machine, we can normalize the vector  $\mathbf{y}'$  to obtain a vector  $\hat{\mathbf{y}}'$  such that  $\|\hat{\mathbf{y}}'\|_2 = (1 \pm \varepsilon_{\text{mach}} \text{poly}(d))$  and  $\|\hat{\mathbf{y}}' - \mathbf{y}'/\|\mathbf{y}'\|_2\|_2 \leq \varepsilon_{\text{mach}} \text{poly}(d)$ . By triangle inequality, we then obtain  $\|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\|_2 \leq \text{poly}(\varepsilon/d) + \varepsilon_{\text{mach}} \text{poly}(d)$ . Finally, for  $i > r$

$$|\langle u_i, \hat{\mathbf{y}}' \rangle| \leq |\langle u_i, \hat{\mathbf{y}} \rangle| + \|\hat{\mathbf{y}} - \hat{\mathbf{y}}'\|_2 \leq \text{poly}(\varepsilon_{\text{pca}}/d) + \varepsilon_{\text{mach}} \text{poly}(d)$$

which then implies that as long as  $\varepsilon_{\text{mach}} \leq \text{poly}(\varepsilon_{\text{pca}}/d)$ , we get  $\sum_{i=r+1}^d \langle u_i, \hat{\mathbf{y}} \rangle^2 \leq \text{poly}(\varepsilon_{\text{pca}})$ .

Thus, we overall obtain that if  $\varepsilon_{\text{mach}} \leq \text{poly}(\varepsilon_{\text{pca}}\eta/d\kappa)$ , we can obtain a vector  $\hat{\mathbf{y}}'$  by running the Lanczos method for  $O(\frac{1}{\sqrt{\varepsilon}} \text{poly}(\log(d/\varepsilon_{\text{pca}}\eta\varepsilon)))$  iterations such that with probability  $\geq 1 - \eta$ ,  $\|\hat{\mathbf{y}}'\|_2 = (1 \pm \varepsilon_{\text{mach}} \text{poly}(d))$  and

$$\sum_{i: \lambda_i(M_s) \leq (1-\varepsilon)\lambda_1(M_s)} \langle \hat{\mathbf{y}}', u_i \rangle^2 \leq \varepsilon_{\text{pca}}.$$

Overall, the algorithm uses  $O(\frac{1}{\sqrt{\varepsilon}} \text{poly}(\log(d/\varepsilon\eta\varepsilon)))$  matrix vector products with  $M_s$  and uses an additional  $O(\frac{d}{\sqrt{\varepsilon}} \text{poly}(\log(d/\varepsilon_{\text{pca}}\eta\varepsilon)))$  floating point operations. ◀

Finally, we modify the LazySVD algorithm (see Algorithm 5) to make it more stable when implemented on a floating point machine. The modification preserves the semantics of the algorithm in the real number model while allowing the stability analysis to go through. For the matrices that we need to run the routine `AppxPCA` on, we show that we can compute very accurate matrix-vector products so that the Lanczos algorithm can be used to approximate the top eigenvector to obtain the following theorem:

► **Theorem 11.** *Given an  $n \times d$  matrix  $A$  with condition number  $\kappa(A) = \sigma_1(A)/\sigma_{k+1}(A)$ , an accuracy parameter  $\varepsilon$ , a rank parameter  $k$  and probability parameter  $\eta$ , if  $\varepsilon_{\text{mach}} \leq \text{poly}(\varepsilon\eta/n\kappa(A))$ , there is an algorithm that outputs a  $d \times k$  matrix  $V_k$  such that  $\kappa(V_k) \leq 4$  and for all  $p \in [2, \infty]$ ,*

$$\|A(I - \text{Proj}_{\text{colspace}(V_k)})\|_{S_p} \leq (1 + O(\varepsilon))\|A - A_k\|_{S_p}$$

and runs in time  $O(\frac{\text{nnz}(A)k}{\sqrt{\varepsilon}} \text{poly}(\log(d\kappa(A)/\varepsilon\eta)) + d \text{poly}(k, \log(dk/\eta\varepsilon)))$ .

### ■ Algorithm 5 Modified LazySVD.

---

**Input:** A positive semidefinite matrix  $M \in \mathbb{R}^{d \times d}$ ,  $k \leq d$ ,  $\varepsilon, \varepsilon_{\text{pca}}, \eta$

**Output:** Vectors  $v'_1, \dots, v'_k$

1  $M_0 \leftarrow M$  and  $V_0 \leftarrow []$

2 **for**  $s = 1, \dots, k$  **do**

3      $v'_s \leftarrow \text{AppxPCA}_{\varepsilon, \varepsilon_{\text{pca}}, \eta/k}((I - \text{Proj}_{\text{colspace}(V_{s-1})})M(I - \text{Proj}_{\text{colspace}(V_{s-1})}))$

4      $V_s \leftarrow [V_{s-1} \ v'_s]$

5 **return**  $V_k$

---

For convenience, we denote any algorithm that satisfies Definition 9 as `AppxPCA` $_{\varepsilon, \varepsilon_{\text{pca}}, \eta}$ . We abuse notation and say that if a unit vector  $w$  satisfies  $\sum_{i \in [d]: \lambda_i(M) \leq (1-\varepsilon)\lambda_1(M)} \langle w, u_i(M) \rangle^2 \leq \varepsilon_{\text{pca}}$ , then “ $w$  is `AppxPCA` $_{\varepsilon, \varepsilon_{\text{pca}}}(M)$ ”.



In [1], the authors show that if  $\varepsilon_{\text{pca}} = \text{poly}(\varepsilon, 1/d, \lambda_{k+1}/\lambda_1)$ , then with probability  $\geq 1 - \eta$  (union bounding over the success of all  $k$  calls to the **AppxPCA** routine), the orthonormal matrix  $V_k$  output by Algorithm 4 satisfies

1.  $\|(I - V_k V_k^\top)M(I - V_k V_k^\top)\|_2 \leq \frac{\lambda_{k+1}(M)}{(1-\varepsilon)}$ ,
2.  $(1 - \varepsilon)\lambda_k(M) \leq v_k^\top M v_k \leq \frac{1}{1-\varepsilon}\lambda_k(M)$  and
3. for every  $p \geq 1$ ,  $\|(I - V_k V_k^\top)M(I - V_k V_k^\top)\|_{S_p} \leq (1 + O(\varepsilon))(\sum_{i=k+1}^d \lambda_i^p)^{1/p}$ .

Since, the modified algorithm (Algorithm 5) has the same semantics as Algorithm 4, the properties 1 and 3 continue to hold for the modified LazySVD algorithm.

The advantage of the modification is that given any vector  $x$ , we can compute  $(I - \text{Proj}_{\text{colspace}(V_s)})x$  very accurately on a floating point machine using stable algorithms for the least squares problem, thereby obtaining a vector  $y$  on a floating point machine that is a very good approximation to  $M_s x = (I - \text{Proj}_{\text{colspace}(V_s)})M((I - \text{Proj}_{\text{colspace}(V_s)}))x$  for any given  $x$ . Below we have a result that states the stability of solving the Least Squares problem on a floating point machine.

► **Theorem 12** (Theorem 19.1 of [15]). *The algorithm for solving the least squares problem  $\min_x \|Ax - b\|_2^2$  using Householder triangulation is backwards stable in the sense that the solution  $\tilde{x}$  satisfies*

$$\|(A + \delta A)\tilde{x} - b\|_2^2 = \min_x \|(A + \delta A)x - b\|_2^2$$

for some matrix  $\delta A$  satisfying  $\|\delta A\|_2 \leq O(\varepsilon_{\text{mach}}\|A\|_2)$ .

Let  $x^* = A^+b$  and from the above theorem, we have  $\tilde{x} = (A + \delta A)^+b$ . Assuming  $\varepsilon_{\text{mach}} \leq 1/2\kappa(A)$ , we have  $A + \delta A$  is full rank and therefore  $(A + \delta A)^+ = ((A + \delta A)^\top(A + \delta A))^{-1}(A + \delta A)^\top$  using which we obtain that  $\|A\tilde{x} - Ax^*\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(\kappa(A))\|b\|_2)$ . Note that  $Ax^* = \text{Proj}_{\text{colspace}(A)}b$ . Thus, given a matrix  $A$  and a vector  $x$ , we can compute a vector  $y$  on a floating point machine such that  $\|y - \text{Proj}_{\text{colspace}(A)}x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(\kappa(A), d)\|x\|_2)$ .

Finally, we can compute another vector  $y'$  satisfying  $\|y' - (I - \text{Proj}_{\text{colspace}(A)})x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(\kappa(A), d)\|x\|_2)$ . Thus, given any vector  $x$ , if operations are computed using machine precision  $\varepsilon_{\text{mach}}$  and if we assume that for any arbitrary vector  $x$ , we can compute a vector  $y$  satisfying  $\|y - Mx\|_2 \leq \varepsilon_M\|M\|_2\|x\|_2$ , then given any vector  $x$ , we can compute a vector  $y$  on a floating point machine satisfying  $\|y - M_s x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(\kappa(V_s), d) + \varepsilon_M)\|M\|_2\|x\|_2$ .

We now bound  $\kappa(V_s)$ . Assume that the vector  $v'_s$  satisfies  $\|v'_s\|_2 = (1 \pm \text{poly}(d)\varepsilon_{\text{mach}})$ . If the vector  $v'_s$  is **AppxPCA** $_{\varepsilon, \varepsilon_{\text{pca}}}(M_{s-1})$ , then

$$\|\text{Proj}_{\text{colspace}(V_{s-1})}v'_s\|_2^2 \leq \sum_{i \in [d]: \lambda_i(M_{s-1}) \leq (1-\varepsilon)\lambda_1(M_{s-1})} \langle v'_s, u_i(M_s) \rangle^2 \leq \varepsilon_{\text{pca}}$$

where the first inequality follows from the fact that  $\text{colspace}(V_{s-1})$  is spanned by the eigenvectors of  $M_{s-1}$  corresponding to zero eigenvalues. Using the above inequality, we can upper bound  $\sigma_{\max}(V_s)$  and lower bound  $\sigma_{\min}(V_s)$ .

► **Lemma 13.** *Suppose  $V_{s-1}$  is a  $d \times (s-1)$  matrix such that  $\sigma_{\max}(V_{s-1}) = \alpha_{s-1}$  and  $\sigma_{\min}(V_{s-1}) = \beta_{s-1}$ . Let  $v'_s$  be a vector with  $\|v'_s\|_2 = (1 \pm \text{poly}(d)\varepsilon_{\text{mach}})$  and satisfies  $\|\text{Proj}_{\text{colspace}(V_{s-1})}v'_s\|_2^2 \leq \varepsilon_{\text{pca}}$ . Let  $V_s = [V_{s-1} \ v'_s]$ . Then  $\sigma_{\max}(V_s) \leq \max(\sigma_{\max}(V_{s-1}), 1 + \text{poly}(d)\varepsilon_{\text{mach}}) + \sqrt{\varepsilon_{\text{pca}}}$  and*

$$\sigma_{\min}(V_s) \geq \sqrt{\max(0, \min(\sigma_{\min}(V_{s-1})^2, 1 - \text{poly}(d)\varepsilon_{\text{mach}}) - \sigma_{\max}(V_{s-1})\sqrt{\varepsilon_{\text{pca}}})}$$

**Proof.** Let  $Q$  be an orthonormal basis for the column space of  $V_{s-1}$  and let  $V_{s-1} = QR$  for a matrix  $R$  with  $\sigma_{\max}(R) = \sigma_{\max}(V_{s-1}) = \alpha_{s-1}$  and  $\sigma_{\min}(R) = \sigma_{\min}(V_{s-1}) = \beta_{s-1}$ . We have that  $\|Q^\top v'_s\|_2^2 = \|QQ^\top v'_s\|_2^2 = \|\text{Proj}_{\text{colspace}(V_{s-1})} v'_s\|_2^2 \leq \varepsilon_{\text{pca}}$ . Let  $x \in \mathbb{R}^s$  be an arbitrary unit vector. Let  $x_1 \in \mathbb{R}^{s-1}$  and  $x_2 \in \mathbb{R}$  be such that  $x = (x_1, x_2)$ . Now,

$$\begin{aligned} \|V_s x\|_2^2 &= \|QRx_1 + v'_s x_2\|_2^2 = \|Rx_1\|_2^2 + x_2^2 \|v'_s\|_2^2 + (2x_2)x_1^\top R^\top Q^\top v'_s \\ &\leq \alpha_{s-1}^2 \|x_1\|_2^2 + (1 + \text{poly}(d)\varepsilon_{\text{mach}})x_2^2 + (2|x_2|)\alpha_{s-1}\|x_1\|_2\sqrt{\varepsilon_{\text{pca}}} \\ &\leq \max(\alpha_{s-1}^2, 1 + \text{poly}(d)\varepsilon_{\text{mach}}) + \alpha_{s-1}\sqrt{\varepsilon_{\text{pca}}}(\|x_1\|_2^2 + |x_2|^2) \end{aligned}$$

which implies that  $\|V_s\|_2 \leq \max(\alpha_{s-1}, 1 + \text{poly}(d)\varepsilon_{\text{mach}}) + \sqrt{\varepsilon_{\text{pca}}}$ . Similarly,

$$\begin{aligned} \|V_s x\|_2^2 &= \|QRx_1 + v'_s x_2\|_2^2 = \|Rx_1\|_2^2 + x_2^2 \|v'_s\|_2^2 + (2x_2)x_1^\top R^\top Q^\top v'_s \\ &\geq \beta_{s-1}^2 \|x_1\|_2^2 + (1 - \text{poly}(d)\varepsilon_{\text{mach}})x_2^2 - 2|x_2|\|x_1\|_2\alpha_{s-1}\sqrt{\varepsilon_{\text{pca}}} \\ &\geq \min(\beta_{s-1}^2, 1 - \text{poly}(d)\varepsilon_{\text{mach}}) - \alpha_{s-1}\sqrt{\varepsilon_{\text{pca}}}. \end{aligned}$$

Hence,  $\sigma_{\min}(V_s) \geq \sqrt{\max(0, \min(\sigma_{\min}(V_{s-1})^2, 1 - \text{poly}(d)\varepsilon_{\text{mach}}) - \sigma_{\max}(V_{s-1})\sqrt{\varepsilon_{\text{pca}}})}$ .  $\blacktriangleleft$

Conditioned on the event that  $\|\text{Proj}_{\text{colspace}(V_{s-1})} v'_s\|_2^2 \leq \varepsilon_{\text{pca}}$  and  $\|v'_s\|_2^2 = (1 \pm \text{poly}(d)\varepsilon_{\text{mach}})$  for all  $s = 1, \dots, k$ , from the above lemma, we obtain that  $\|V_s\|_2 \leq 1 + \text{poly}(d)\varepsilon_{\text{mach}} + k\sqrt{\varepsilon_{\text{pca}}}$ . If  $\varepsilon_{\text{pca}} \leq 1/\text{poly}(k)$  and  $\varepsilon_{\text{mach}} \leq 1/\text{poly}(d)$ , then  $\|V_s\|_2 \leq 2$  for all  $s = 1, \dots, k$  which in turn implies that for all  $s = 1, \dots, k$ ,  $\sigma_{\min}(V_s) \geq \sqrt{1 - \text{poly}(d)\varepsilon_{\text{mach}} - 2k\sqrt{\varepsilon_{\text{pca}}}} \geq 1/2$  assuming  $\varepsilon_{\text{pca}} \leq 1/\text{poly}(k)$  and  $\varepsilon_{\text{mach}} \leq 1/\text{poly}(d)$ .

Hence,  $\kappa(V_s) \leq 4$  for all  $s = 1, \dots, k$  in Algorithm 5 conditioned on the success of all the calls to **AppxPCA**. Thus, we can assume that given any vector  $x$ , we can compute a vector  $y$  on a floating point computer with precision  $\varepsilon_{\text{mach}} \leq 1/\text{poly}(d)$  such that  $\|y - M_s x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(d) + \varepsilon_M)\|M\|_2\|x\|_2$ .

Let  $A \in \mathbb{R}^{n \times d}$  be an arbitrary matrix with  $n \geq d$ . Define  $M = A^\top A$  to be a  $d \times d$  matrix. Given any vector  $x$ , we can compute a vector  $y$  satisfying  $\|A^\top A x - y\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(n)\|A\|_2^2\|x\|_2)$ . As  $\|A\|_2^2 = \|M\|_2$ , we thus have that for any  $x$ , we can compute  $y$  satisfying  $O(\varepsilon_{\text{mach}} \text{poly}(d)\|M\|_2\|x\|_2)$ . Thus,  $\varepsilon_M$  as defined above can be taken as  $\varepsilon_{\text{mach}} \text{poly}(d)$ . Let  $\kappa = \lambda_1(M)/\lambda_{k+1}(M)$ . By definition of eigenvalues, for any matrix  $V$  with at most  $k$  columns, we have  $\|(I - \text{Proj}_{\text{colspace}(V)})M(I - \text{Proj}_{\text{colspace}(V)})\|_2 \geq \lambda_{k+1}$ . Hence for all  $s = 1, \dots, k$ ,  $\|M_s\|_2 \geq \|M\|_2/\kappa$ . Thus, given any vector  $x$ , we can compute a vector  $y$  satisfying  $\|y - M_s x\|_2 \leq O(\varepsilon_{\text{mach}} \text{poly}(n)\kappa)\|M_s\|_2\|x\|_2$ .

Finally, we have the main theorem showing the stability of the LazySVD algorithm.

**Proof of Theorem 11.** Note that the algorithm in Lemma 10 satisfies the **AppxPCA** $_{\varepsilon, \varepsilon_{\text{pca}}, \eta}$  definition. Thus, if  $\varepsilon_{\text{pca}} \leq \text{poly}(\varepsilon/d\kappa)$ , by Theorem 4.1 of [1], Algorithm 5 when run on the  $d \times d$  matrix  $A^\top A$  outputs a matrix  $V_k$  such that with probability  $\geq 1 - \eta$ ,  $\|(I - \text{Proj}_{\text{colspace}(V_k)})A^\top A(I - \text{Proj}_{\text{colspace}(V_k)})\|_2 \leq \frac{1}{1-\varepsilon}\sigma_{k+1}(A)^2$  and for all  $p' \geq 1$ ,

$$\|(I - \text{Proj}_{\text{colspace}(V_k)})A^\top A(I - \text{Proj}_{\text{colspace}(V_k)})\|_{S_{p'}} \leq (1 + O(\varepsilon))\left(\sum_{i=k+1}^d \sigma_i(A)^{2p'}\right)^{1/p'}.$$

Thus, we have  $\|A(I - \text{Proj}_{\text{colspace}(V_k)})\|_2 \leq (1 + O(\varepsilon))\sigma_{k+1}(A)$  and using the fact that  $\|A^\top A\|_{S_p}^p = \|A\|_{S_{2p}}^{2p}$ , for all  $p \geq 2$ ,  $\|A(I - \text{Proj}_{\text{colspace}(V_k)})\|_{S_p} \leq (1 + O(\varepsilon))\|A - A_k\|_p$ . We additionally have  $\kappa(V_k) \leq 4$  from Lemma 13.

**Runtime Analysis.** In each iteration of Algorithm 5, we require  $O(\varepsilon^{-1/2} \text{poly}(\log(d\kappa/\eta\varepsilon)))$  matrix vector products with the matrices  $A$  and  $A^\top$ . For iterations  $s = 1, \dots, k$ , we solve  $O(\varepsilon^{-1/2} \text{poly}(\log(d\kappa/\eta\varepsilon)))$  least squares problems on a fixed  $d \times s$  matrix and different label vectors. Thus, the overall time complexity of the algorithm is

$$O\left(\frac{\text{nnz}(A)k}{\sqrt{\varepsilon}} \text{poly}(\log(d\kappa/\eta\varepsilon)) + d \text{poly}(k, \log(d\kappa/\eta\varepsilon))\right). \quad \blacktriangleleft$$

---

## References

- 1 Zeyuan Allen-Zhu and Yuanzhi Li. LazySVD: Even faster SVD decomposition yet without agonizing pain. *Advances in neural information processing systems*, 29, 2016. URL: <https://proceedings.neurips.cc/paper/2016/hash/c6e19e830859f2cb9f7c8f8cacb8d2a6-Abstract.html>.
- 2 Ainesh Bakshi, Kenneth L Clarkson, and David P Woodruff. Low-rank approximation with  $1/\varepsilon^{1/3}$  matrix-vector products. *STOC 2022. arXiv:2202.05120*, 2022. doi:10.48550/arXiv.2202.05120.
- 3 Jess Banks, Jorge Garza-Vargas, Archit Kulkarni, and Nikhil Srivastava. Pseudospectral shattering, the sign function, and diagonalization in nearly matrix multiplication time. *Foundations of Computational Mathematics*, pages 1–89, 2023. doi:10.1007/s10208-022-09577-5.
- 4 Kenneth L. Clarkson and David P. Woodruff. Low-rank approximation and regression in input sparsity time. *J. ACM*, 63(6):Art. 54, 45, 2017. doi:10.1145/3019134.
- 5 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. *FOCS 2023. arXiv:2210.10173*, 2022. doi:10.48550/arXiv.2210.10173.
- 6 François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018. doi:10.1137/1.9781611975031.67.
- 7 François Le Gall. Faster algorithms for rectangular matrix multiplication. In *2012 IEEE 53rd annual symposium on foundations of computer science*, pages 514–523. IEEE, 2012. doi:10.1109/FOCS.2012.80.
- 8 François Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3765–3791. SIAM, 2024. doi:10.1137/1.9781611977912.133.
- 9 Yi Li and David Woodruff. Input-sparsity low rank approximation in Schatten norm. In *International Conference on Machine Learning*, pages 6001–6009. PMLR, 2020. URL: <http://proceedings.mlr.press/v119/li20q.html>.
- 10 Grazia Lotti and Francesco Romani. On the asymptotic complexity of rectangular matrix multiplication. *Theoretical Computer Science*, 23(2):171–185, 1983. doi:10.1016/0304-3975(83)90054-3.
- 11 Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Advances in neural information processing systems*, 28, 2015. URL: <https://proceedings.neurips.cc/paper/2015/hash/1efa39bcaec6f3900149160693694536-Abstract.html>.
- 12 Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the Lanczos method for matrix function approximation. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1605–1624. SIAM, Philadelphia, PA, 2018. doi:10.1137/1.9781611975031.105.
- 13 Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9(2):125–210, 2014. doi:10.1561/0400000065.

- 14 Aleksandros Sobczyk, Marko Mladenović, and Mathieu Luisier. Hermitian pseudospectral shattering, cholesky, hermitian eigenvalues, and density functional theory in nearly matrix multiplication time. *arXiv preprint arXiv:2311.10459*, 2023. doi:10.48550/arXiv.2311.10459.
- 15 Lloyd N. Trefethen and David Bau, III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997. doi:10.1137/1.9780898719574.
- 16 Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011. doi:10.1142/S1793536911000787.

## A Time Complexity of SVD in the Real RAM model

Consider an  $m \times n$  matrix  $A$  where  $m \leq n$ . We can compute the matrix  $M = A^\top A$  in time  $O(nm^{\omega-1})$ , where  $\omega$  is the matrix multiplication exponent. Using the eigendecomposition algorithm of [3], we can then compute a matrix  $V$  and a diagonal matrix  $D$  satisfying  $\|M - VDV^{-1}\|_2 \leq \|M\|_2 / \text{poly}(n)$  in time  $\tilde{O}(m^\omega)$ . Although the matrix  $M$  is symmetric, the matrix  $V$  output by the algorithm may not be orthonormal. In the real RAM model, we can perform the following changes to their algorithm:

1. The Ginibre perturbation step is replaced with the symmetric Gaussian Orthogonal Ensemble perturbation as mentioned in Remark 6.1 of [3].
2. In step 5 of the algorithm EIG in [3], after computing the orthonormal matrices  $\tilde{Q}_+$  and  $\tilde{Q}_-$ , we modify  $\tilde{Q}_-$  to an orthonormal basis of the column space of the matrix  $(I - \tilde{Q}_+ \tilde{Q}_+^\top) \tilde{Q}_-$ . This ensures that  $\tilde{Q}_+^\top \tilde{Q}_- = 0$ , while preserving the properties of  $\tilde{Q}_-$  guaranteed by the algorithm DEFLATE. Note that the matrix  $\tilde{Q}_-$  can be updated in time  $\tilde{O}(n^\omega)$  in the real RAM model.

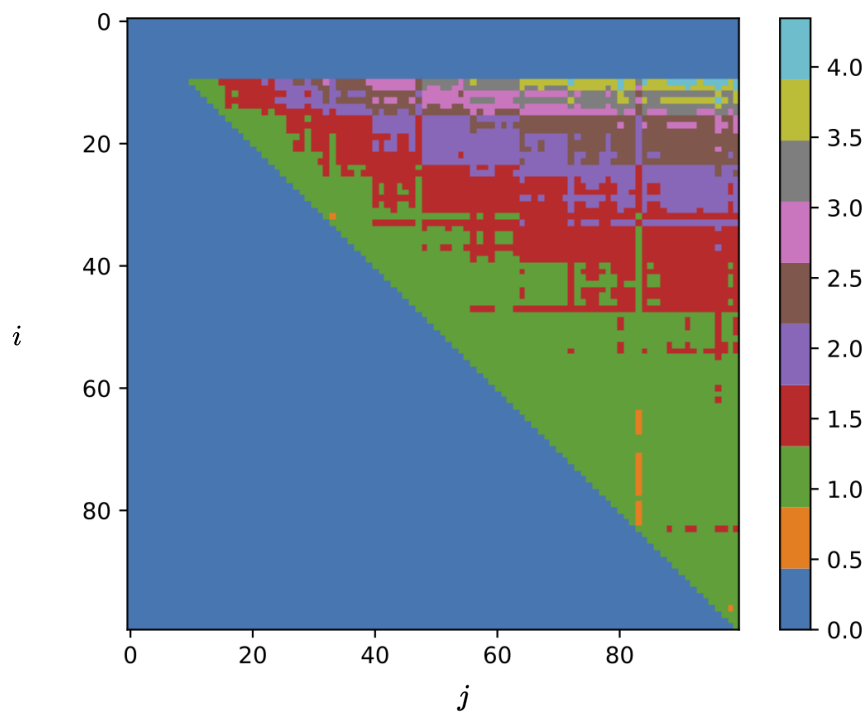
Thus, the algorithm of [3] can be used to compute an orthonormal matrix  $V$  and a diagonal matrix  $D$  such that  $\|M - VDV^\top\|_2 \leq \|M\|_2 / \text{poly}(n)$  in  $\tilde{O}(nm^{\omega-1})$  time in the real RAM model.

If we define  $U = AV \cdot D^{-1/2}$ , then  $U, D^{1/2}, V^\top$  is an approximate singular value decomposition of the matrix  $A$ , where the matrices  $U, V$  are orthonormal up to a  $1/\text{poly}(n)$  error. Since the matrix  $AV$  can be computed in  $\tilde{O}(nm^{\omega-1})$ , we obtain that SVD of a well conditioned matrix can be computed in  $\tilde{O}(nm^{\omega-1})$  time.

## B An Experiment

We consider multiplying an  $n \times n$  matrix with an  $n \times d$  matrix while varying  $d$ . We set  $n = 10,000$  and vary  $d$  to take values in the interval  $[10, 100]$ . If  $t_d$  is the median amount of time (over 5 repetitions) to compute the product of an  $n \times n$  matrix with an  $n \times d$  matrix, we obtain a color map (Figure 1) of the values  $(j/i)/(t_j/t_i)$  for  $j \geq i$ . If  $(j/i)/(t_j/t_i)$  is large then  $t_j$  is much smaller than  $t_i(j/i)$  which is what we would expect if the matrix-multiplication time scales linearly with the dimension.



The experiment was performed using NumPy on a machine with 2 cores. We see that fixing an  $i$ , as we increase  $j$ ,  $t_j$  becomes smaller compared to  $t_i \cdot (j/i)$ . Hence, it is advantageous to run with larger block sizes if it means that it reduces the number of iterations for which the smaller block size is to be run. In the proof of Theorem 5, we see that if we increase the larger block to 4 times the original, then the number of iterations the smaller block size is to be run decreases to 0.5x the original. Based on the characteristics of the machine, we can obtain significant improvements over the parameters obtained by optimizing for matrix-vector products.



■ **Figure 1** Color Map of  $(j/i)/(t_j/t_i)$ .



# Fast and Slow Mixing of the Kawasaki Dynamics on Bounded-Degree Graphs

Aiya Kuchukova  

School of Mathematics, Georgia Institute of Technology, Atlanta, GA, USA

Marcus Pappik  

Hasso Plattner Institute, University of Potsdam, Germany

Will Perkins  

School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA

Corrine Yap  

School of Mathematics, Georgia Institute of Technology, Atlanta, GA, USA

---

## Abstract

We study the worst-case mixing time of the global Kawasaki dynamics for the fixed-magnetization Ising model on the class of graphs of maximum degree  $\Delta$ . Proving a conjecture of Carlson, Davies, Kolla, and Perkins, we show that below the tree uniqueness threshold, the Kawasaki dynamics mix rapidly for all magnetizations. Disproving a conjecture of Carlson, Davies, Kolla, and Perkins, we show that the regime of fast mixing does not extend throughout the regime of tractability for this model: there is a range of parameters for which there exist efficient sampling algorithms for the fixed-magnetization Ising model on max-degree  $\Delta$  graphs, but the Kawasaki dynamics can take exponential time to mix. Our techniques involve showing spectral independence in the fixed-magnetization Ising model and proving a sharp threshold for the existence of multiple metastable states in the Ising model with external field on random regular graphs.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Markov-chain Monte Carlo methods; Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures

**Keywords and phrases** ferromagnetic Ising model, fixed-magnetization Ising model, Kawasaki dynamics, Glauber dynamics, mixing time

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.56

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2405.06209> [36]

**Funding** *Marcus Pappik:* supported by the HPI Research School on Data Science and Engineering.

*Will Perkins:* supported in part by NSF grant CCF-2309708.

*Corrine Yap:* supported in part by NIH grant R01GM126554.

**Acknowledgements** The authors thank Zongchen Chen and Marcus Michelen for very helpful discussions.

## 1 Introduction

The Ising model on a finite graph  $G = (V, E)$  is the following probability distribution on  $\Omega = \{+1, -1\}^V$ :

$$\mu_{G,\beta,\lambda}(\sigma) = \frac{\lambda^{|\sigma|^+} e^{\beta m_G(\sigma)}}{Z_G(\beta, \lambda)} \quad (1)$$

where  $|\sigma|^+ = |\{\sigma^{-1}(+1)\}|$  is the number of vertices assigned a  $+1$  spin under  $\sigma$  which we call the *size* of  $\sigma$ , and  $m_G(\sigma)$  is the number of monochromatic edges in  $G$  under the 2-coloring given by  $\sigma \in \Omega$ . The measure  $\mu_{G,\beta,\lambda}$  is called the *Gibbs measure* on  $G$  with *inverse temperature*



© Aiya Kuchukova, Marcus Pappik, Will Perkins, and Corrine Yap;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 56; pp. 56:1–56:24



Leibniz International Proceedings in Informatics

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

$\beta \geq 0$  and *external field*  $\lambda \geq 0$ . The normalizing constant  $Z_G(\beta, \lambda) = \sum_{\sigma \in \Omega} \lambda^{|\sigma|^+} e^{\beta m_G(\sigma)}$  is the *partition function* of the Ising model. Throughout this paper, we focus on the *ferromagnetic* case,  $\beta \geq 0$ , in which agreeing spins on edges are preferred.

Spin models on graphs are the source of many interesting computational problems. Questions about the tractability of approximate counting (estimating the partition function) and approximate sampling (from the Gibbs distribution) are studied extensively.

In the case of the ferromagnetic Ising model, Jerrum and Sinclair [35] showed that there is a polynomial-time approximation algorithm on all graphs at all temperatures, and Randall and Wilson [42] gave an efficient sampling algorithm.

In other cases, such as the anti-ferromagnetic Ising model ( $\beta < 0$ ) and the hard-core model of weighted independent sets, approximate counting and sampling can be computationally hard (e.g., no polynomial-time algorithm exists unless NP=RP). For the class  $\mathcal{G}_\Delta$  of graphs of maximum degree  $\Delta$ , these two models exhibit *computational thresholds*: as the activity or external field parameter  $\lambda$  varies, there is a sharp threshold between tractability (efficient approximate counting and sampling) and intractability (NP-hardness) [28, 46–48]. Moreover, the critical value  $\lambda_c = \lambda_c(\Delta, \beta)$  is the phase transition point of the corresponding model on the infinite  $\Delta$ -regular tree  $\mathbb{T}_\Delta$  (more precisely, it is the threshold for the *uniqueness of Gibbs measure* on  $\mathbb{T}_\Delta$ , a notion which we discuss shortly). Thus there is a remarkable connection between computational thresholds and statistical physics phase transitions. Even further, the threshold  $\lambda_c$  has recently been shown to be a *dynamical threshold*: it is the threshold for rapid mixing of the Glauber dynamics, a natural Markov chain for sampling from spin models like the Ising or hard-core models, on graphs in  $\mathcal{G}_\Delta$  [2, 15, 41, 48]. So in these cases, three different thresholds (computational, dynamical, uniqueness on the tree) coincide.

A very similar picture has emerged for the model of a uniformly random independent set of a given size. For the class of graphs  $\mathcal{G}_\Delta$ , there is a critical density  $\alpha_c(\Delta)$  so that if  $\alpha < \alpha_c$ , there are efficient algorithms to approximately count and sample independent sets of density  $\alpha$ , while if  $\alpha > \alpha_c$  no such algorithms exist unless NP=RP [22]. Jain, Michelen, Pham, and Vuong [33] recently proved that this computational threshold  $\alpha_c$  also marks the dynamical threshold – for  $\alpha < \alpha_c$ , the natural “down-up” random walk on independent sets of a given size mixes rapidly. The threshold  $\alpha_c(\Delta)$  is closely connected to a uniqueness threshold on the tree: it is the smallest expected density of an independent set in the hard-core model on  $G \in \mathcal{G}_\Delta$  at activity  $\lambda_c(\Delta)$ .

Returning to the ferromagnetic Ising model ( $\beta \geq 0$ ), the picture is fundamentally different and not completely understood. While there is no computational threshold (there are efficient algorithms for all parameters) one can still ask about the relationship between uniqueness and dynamical thresholds. The natural dynamics in this setting are the *Glauber dynamics*, a Markov chain on the state space  $\Omega$  with stationary distribution  $\mu_{G, \beta, \lambda}$  which at each step chooses a uniformly random vertex and updates its spin according to the conditional distribution given the spins of its neighbors. For the case  $\lambda = 1$  (“no external field”) the dynamical threshold has been identified, and it coincides with the uniqueness threshold. For  $\Delta \geq 3$ , let the *critical inverse temperature* of the Ising model on  $\mathbb{T}_\Delta$  be denoted by

$$\beta_u(\Delta) := \ln \left( \frac{\Delta}{\Delta - 2} \right).$$

The value  $\beta_u(\Delta)$  is the Gibbs uniqueness threshold for the Ising model (with  $\lambda = 1$ ) on  $\mathbb{T}_\Delta$  (see e.g. [6] and below in Section 2.1 for a precise definition). Mossel and Sly [40] proved that for  $0 \leq \beta < \beta_u$  and any  $\lambda$ , the Glauber dynamics are rapidly mixing for any  $G \in \mathcal{G}_\Delta$ . This threshold in  $\beta$  is sharp due to the analysis of the random  $\Delta$ -regular graph in [23, 31]: for  $\beta > \beta_u$  and  $\lambda = 1$ , the Glauber dynamics for the Ising model take exponential time to mix.



For general  $\lambda \geq 0$ , in the regime  $\beta > \beta_u$ , the threshold landscape is not as well understood. Note that the model is symmetric around  $\lambda = 1$  by swapping the role of  $+$  and  $-$  spins and so for each threshold, its inverse is also a threshold; for clarity we will define thresholds for the case  $\lambda \geq 1$ . Let  $\lambda_u(\Delta, \beta)$  be the Gibbs uniqueness threshold of the ferromagnetic Ising model on  $\mathbb{T}_\Delta$ ; that is,  $\lambda_u$  is the smallest  $\lambda_0 \geq 1$  so that there is a unique Gibbs measure for the Ising model on  $\mathbb{T}_\Delta$  with inverse temperature  $\beta$  and external field  $\lambda$ , for all  $\lambda > \lambda_0$  (again see [6] and Section 2.1 for details). The value of  $\lambda_u$  can be given implicitly as the solution to an equation involving  $\Delta, \beta$ , and  $\lambda$ . Unlike in the above mentioned examples, while  $\lambda_u$  marks a phase transition on the tree, it does not mark a computational transition (since sampling from the ferromagnetic Ising model is tractable on all graphs and all parameters) and it has not been established as a dynamical threshold (though this also has not been ruled out). Below in Theorem 2 we show that the worst-case mixing time of Glauber dynamics over  $\mathcal{G}_\Delta$  is exponential when  $|\log \lambda| < \log \lambda_u$ .

The complementary result (fast mixing of the Glauber dynamics for  $G \in \mathcal{G}_\Delta$  when  $|\log \lambda| > \log \lambda_u$ ) is not known to hold. Instead, sufficient conditions for fast mixing have been given that require  $\lambda$  to be somewhat larger than  $\lambda_u$ . An interesting insight is that upper bounds on the dynamical threshold are often connected to zero-freeness of the map  $\lambda \mapsto Z_G(\beta, \lambda)$  considered as a complex polynomial. Throughout this paper, we particularly focus on the *analytic threshold*  $\lambda_a(\Delta, \beta)$ , defined by the following requirement: for all  $G \in \mathcal{G}_\Delta$ , every compact  $D \subset (\lambda_a(\Delta, \beta), \infty)$  and every partial spin assignment  $\tau_U : U \rightarrow \{-1, +1\}$ ,  $U \subset V$  it holds that  $Z_G^{\tau_U}(\beta, \lambda)$  (the partition function restricted to configurations that are consistent with  $\tau_U$ ) is non-zero for all  $\lambda$  in some uniform complex neighborhood of  $D$ . A formal definition of  $\lambda_a$  is given in Section 2.5. In contrast to the uniqueness threshold,  $\lambda_a(\Delta, \beta)$  has not been determined. It is known that  $\lambda_a(\Delta, \beta) \geq \lambda_u(\Delta, \beta)$  and the best known upper bound is

$$\lambda_a(\Delta, \beta) \leq \min \left\{ \frac{(\Delta - 2)e^{2\beta} - \Delta}{e^{\beta(2-\Delta)}}, e^{\beta\Delta} \right\} =: \bar{\lambda}_a. \tag{2}$$

The first expression in the minimum of (2) was proven by Shao and Sun [44], and the second bound of  $e^{\beta\Delta}$  (which is smaller than the first expression for  $\Delta \geq 4$  and  $\beta$  large enough) was proven by Shao and Ye [45].

It turns out that this analytic threshold  $\lambda_a$  is closely related to the dynamical threshold. More precisely, Chen, Liu, and Vigoda [17] proved that the first bound in (2) can be used to define a regime in which the ferromagnetic Ising model satisfies  $\ell_\infty$ -independence (see Section 2.4), a stronger version of spectral independence that implies rapid mixing of Glauber dynamics. Their derivation of the threshold used techniques similar to those of Shao and Sun [44] which resulted in coinciding bounds, but a more systematic connection was provided by Chen, Liu and Vigoda in [16]. They showed that for a broad class of spin systems, sufficiently strong zero-freeness assumptions imply  $\ell_\infty$ -independence. With small adjustments, we use their technique to argue that the ferromagnetic Ising model satisfies  $\ell_\infty$ -independence for all  $|\log \lambda| > \log \lambda_a(\Delta, \beta)$  (see Theorem 22).

The main focus of this paper is on dynamical thresholds of the *fixed-magnetization* Ising model with inverse temperature  $\beta$  and magnetization  $\eta$ . The magnetization (per vertex) of an Ising configuration  $\sigma$  is  $\eta(\sigma) := \frac{\sum_{v \in V(G)} \sigma_v}{|V(G)|}$ . A configuration  $\sigma$  of magnetization  $\eta$  has size (number of  $+1$  spins) exactly  $k = \lfloor n \frac{\eta+1}{2} \rfloor$ . We denote by  $\Omega_k$  the configurations of size  $k$ .

The fixed-magnetization Ising model with inverse temperature  $\beta \geq 0$  and magnetization  $\eta \in [-1, 1]$  is then a probability distribution defined similarly to (1) but on  $\Omega_k$ , where  $k = \lfloor n \frac{\eta+1}{2} \rfloor$ , as

$$\hat{\mu}_{G,\beta,\eta}(\sigma) = \frac{e^{\beta m_G(\sigma)}}{\hat{Z}_{G,\eta}(\beta)},$$

where

$$\hat{Z}_{G,\eta}(\beta) = \sum_{\sigma \in \Omega_k} e^{\beta m_G(\sigma)}$$

is the fixed-magnetization partition function. Here we use floors to avoid restricting to values of  $\eta$  where  $n \frac{\eta+1}{2}$  is an integer. The distribution  $\hat{\mu}_{G,\beta,\eta}$  is exactly that of  $\mu_{G,\beta,\lambda}$  conditioned on the event  $\{\sigma \in \Omega_k\}$ . Note that the external field plays no role in the fixed-magnetization model since  $\lambda^{|\sigma|^+}$  is constant on  $\Omega_k$ .

In statistical physics, the fixed-magnetization Ising model is the *canonical ensemble* while the Ising model is the *grand canonical ensemble*. The fixed-magnetization model on lattices is studied in, e.g., [13, 24], where interesting geometric behavior is described; the behavior of the Kawasaki dynamics (the natural analogue of Glauber dynamics) on  $\mathbb{Z}^d$  has been studied extensively in, e.g., [9–11, 38]. Here we focus on dynamical behavior over the class of all graphs of maximum degree  $\Delta$ .

To understand algorithmic and dynamical thresholds in the fixed-magnetization Ising model, we need to define some further parameters. The mean magnetization of the  $+$  measure on  $\mathbb{T}_\Delta$  (explained in detail in Section 2.1) is

$$\eta_{\Delta,\beta,\lambda}^+ := \tanh(L^* + \operatorname{artanh}(\tanh(L^*) \tanh(\beta/2)))$$

where  $L^*$  is the largest solution to

$$L = \log(\lambda) + (\Delta - 1) \operatorname{artanh}(\tanh(L) \tanh(\beta/2)).$$

We are specifically interested in the following three quantities:

$$\eta_c(\Delta, \beta) = \eta_{\Delta,\beta,1}^+ \quad \eta_u(\Delta, \beta) = \eta_{\Delta,\beta,\lambda_u}^+ \quad \eta_a(\Delta, \beta) = \eta_{\Delta,\beta,\lambda_a}^+.$$

For  $\beta > \beta_u$ , we have  $0 < \eta_c < \eta_u \leq \eta_a$ . It is not known if the last inequality is strict or not (just as it is not known if  $\lambda_a = \lambda_u$ ).

Carlson, Davies, Kolla, and Perkins [12] showed recently that the fixed-magnetization Ising model exhibits quite different algorithmic behavior than the Ising model: it exhibits a computational threshold. In particular, for  $\beta < \beta_u$  and any  $\eta$ , as well as for  $\beta > \beta_u$  and  $|\eta| > \eta_c$ , there are efficient approximate counting and sampling algorithms for the Ising model at fixed mean magnetization  $\eta$  on  $\mathcal{G}_\Delta$ , while for  $\beta > \beta_u$  and  $|\eta| < \eta_c$ , there are no such algorithms unless NP=RP. Thus  $\beta_u$  and  $\eta_c$  mark the computational threshold in the fixed-magnetization Ising model.

Here we study dynamical thresholds for the fixed-magnetization Ising model on  $\mathcal{G}_\Delta$ . Given a distribution, one candidate for an efficient approximate sampling algorithm is a Markov chain whose stationary distribution is our target distribution, but the efficiency of this algorithm depends on the mixing time. Recall that the mixing time of a Markov chain is the number of steps, in the worst-case over initial distribution, required for a Markov chain to reach  $1/4$  total variation distance of its stationary distribution (see Section 2.4 for a formal definition). As mentioned above, the natural dynamics associated to the fixed-magnetization

Ising model are the *Kawasaki dynamics*, which is a reversible Markov chain on  $\Omega_k$ . At each step of the chain, a  $+1$  vertex and a  $-1$  vertex are chosen uniformly at random and have their spins swapped with a probability depending on the ratio of the Ising probabilities of the two configurations. This is sometimes referred to as the *global* Kawasaki dynamics, whereas the *local* Kawasaki dynamics restrict to swapping spins of neighboring vertices.

Our main contributions concern the *mixing time* of the Kawasaki dynamics. Taking  $\|\mu - \nu\|_{\text{TV}} := \sup_{A \in \mathcal{A}} |\mu(A) - \nu(A)|$  to be the total variation distance between probability distributions  $\mu$  and  $\nu$  on a probability space  $(\Omega, \mathcal{A})$ , the mixing time of a Markov chain on  $\Omega$  that has transition matrix  $P$  and stationary distribution  $\pi$  is

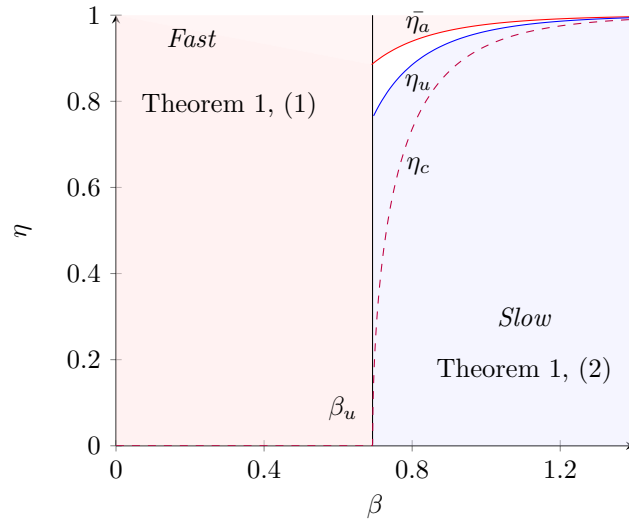
$$\tau_{\text{mix}} := \inf \left\{ t : \max_{x \in \Omega} \|P^t(x, \cdot) - \pi\|_{\text{TV}} \leq \frac{1}{4} \right\}.$$

Resolving one conjecture of Carlson, Davies, Kolla, and Perkins and disproving another (part (i) and (ii) respectively of [12, Conjecture 1]), we establish thresholds in the mean magnetization for fast and slow mixing of the Kawasaki dynamics on  $\mathcal{G}_\Delta$ .

► **Theorem 1.** *For the Kawasaki dynamics, the following two statements hold:*

- (1) *If  $0 \leq \beta < \beta_u$  or if  $\beta > \beta_u$  and  $|\eta| > \eta_a$ , then the Kawasaki dynamics for  $\hat{\mu}_{G, \beta, \eta}$  have mixing time  $O(|V(G)|^2)$  for all  $G \in \mathcal{G}_\Delta$ .*
- (2) *There exists a sequence of graphs  $G_n \in \mathcal{G}_\Delta$  with  $|V(G_n)| \rightarrow \infty$  such that for  $\beta > \beta_u$  and  $|\eta| < \eta_u$ , the Kawasaki dynamics for  $\hat{\mu}_{G, \beta, \eta}$  have mixing time  $\exp(\Omega(|V(G_n)|))$  on  $G$ .*

Fast mixing of the dynamics for all  $\eta$  when  $\beta < \beta_u$  was conjectured in [12]. The slow mixing for some  $\eta > \eta_c$  disproves the conjecture from [12] asserting the coincidence of the algorithmic and dynamical thresholds. If it were established that  $\lambda_a(\Delta, \beta) = \lambda_u(\Delta, \beta)$  then Theorem 1 would give the sharp dynamical threshold for the fixed-magnetization model. It is an interesting question to understand the dynamical threshold in both the Ising model and fixed-magnetization Ising model if instead it holds that  $\lambda_u < \lambda_a$ .



■ **Figure 1** Sketch of the phase space for the fixed-magnetization model on  $\mathcal{G}_\Delta$  when  $\Delta = 4$ , where  $\bar{\eta}_a = \eta_{\Delta, \beta, \bar{\lambda}_a}$ .

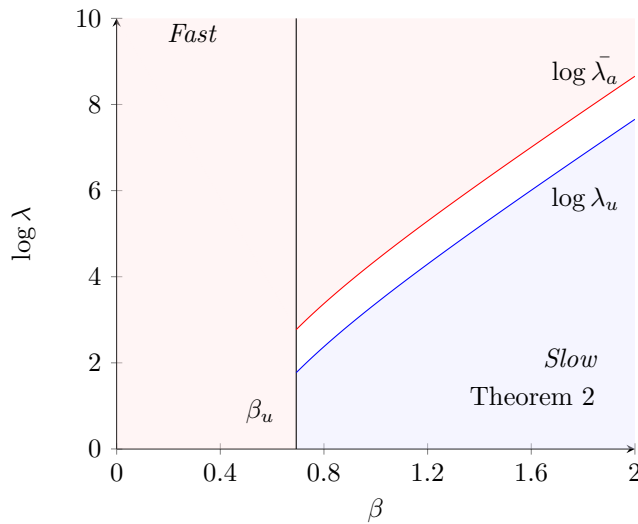
A diagram of the computational and dynamical thresholds for the fixed-magnetization Ising model is given in Figure 1.

Towards the proof of Theorem 1,(2), we establish that the Glauber dynamics for the Ising model on the random  $\Delta$ -regular graph takes exponential time to mix when  $\beta > \beta_u$  and  $\lambda$  is in the non-uniqueness regime for  $\mathbb{T}_\Delta$ .

► **Theorem 2.** Fix  $\Delta \geq 3$ ,  $\beta > \beta_u(\Delta)$ , and  $|\log \lambda| < \log \lambda_u(\Delta, \beta)$ . Let  $G$  be a uniformly random  $\Delta$ -regular graph on  $n$  vertices. Then with high probability as  $n \rightarrow \infty$ , the mixing time of the Glauber dynamics for the Ising model on  $G$  is  $e^{\Theta(n)}$ .

This theorem complements the result of Can, van der Hofstad, and Kumagai [8] showing that when  $|\log \lambda| > \log \lambda_u$ , with high probability over the random regular graph the mixing time of the Glauber dynamics is  $O(n \log n)$ ; they conjectured that the mixing time is exponential when  $|\log \lambda| < \log \lambda_u$ , which Theorem 2 confirms.

Theorem 2 also fills in more of the picture for dynamical thresholds in the Ising model on graphs in  $\mathcal{G}_\Delta$ ; see Figure 2.



■ **Figure 2** Sketch of the phase space for the Ising model Glauber dynamics on  $\mathcal{G}_\Delta$  when  $\Delta = 4$ .

Before we give an overview of our proof techniques, we state some open questions. Our first question is concerned with the relation between the analytic threshold and the uniqueness threshold for the Ising model.

► **Question 3.** Does  $\lambda_a(\Delta, \beta) = \lambda_u(\Delta, \beta)$ ?

If the answer is yes, then by the results above we would have a complete characterization of the dynamical thresholds in the Ising and fixed magnetization Ising models on  $\mathcal{G}_\Delta$ .

Next we conjecture the following improvement of part (1) of Theorem 1.

► **Conjecture 4.** If  $0 \leq \beta < \beta_u$  or if  $\beta > \beta_u$  and  $|\eta| > \eta_a$ , then the Kawasaki dynamics for  $\hat{\mu}_{G,\beta,\eta}$  are optimally mixing: the mixing time is in  $O(|V(G)| \cdot \log(|V(G)|))$  for all  $G \in \mathcal{G}_\Delta$ .

The analogous statement for independent sets is proved in [33] by proving a log-Sobolev inequality for the down-up walk with constant  $\Omega(1/n)$ .

While we focus on global Kawasaki dynamics in this paper, we suggest that our results also apply to the local dynamics. Note that for studying local Kawasaki dynamics, it makes sense to assume that  $G$  is connected. In this case, we believe that a Markov chain comparison argument as in [26] can be used to show that the mixing times of the local and global

dynamics only differ by a polynomial factor. While our slow mixing result for the global dynamics uses identical copies of disjoint random graphs, our arguments should still apply if they are connected with a sparse set of edges. As a consequence, both slow and rapid mixing from Theorem 1 would carry over. A full proof of this is left for future work.

## 1.1 Overview of Techniques

The proofs of Theorems 1 and 2 involve several different ingredients, including local central limit theorems, spectral independence, and first- and second-moment methods for spin models on random graph. We give an overview of the techniques here.

### 1.1.1 Fast Mixing

At a high level, the proof of Theorem 1, (1) follows the strategy used by Jain, Michelen, Pham, and Vuong [33] to show fast mixing for the down-up walk on independent sets of density less than  $\alpha_c(\Delta)$ .

In order to derive an upper bound on the mixing time of the Kawasaki dynamics for the fixed-magnetization Ising model, we prove that the spectral gap of the associated transition matrix is bounded below by  $\Omega(1/n)$ . To achieve this, we study a related down-up Ising walk on  $\Omega_k$  while arguing that the respective spectral gaps of the Kawasaki dynamics and the down-up walk are within a constant factor of each other. This allows us to make use of recent literature that relates the spectral gap of a down-up walk to spectral independence [1, 2, 14].

Informally speaking, spectral independence captures the idea that for most pairs of vertices  $v, w \in V$ , the spins assigned to  $v$  and  $w$  by a random configuration from  $\hat{\mu}_{G,\beta,\eta}$  are almost independent. While spectral independence for the Ising model has been studied before by Chen, Liu, and Vigoda [17], no comparable result exists for the fixed-magnetization model. To derive the required spectral independence property, we follow an approach introduced in [33] to analyze the down-up walk for fixed-size independent sets. The idea is to choose  $\lambda$  such that a random configuration from  $\mu_{G,\beta,\lambda}$  has expected magnetization per vertex close to  $\eta$ . We then view  $\hat{\mu}_{G,\beta,\eta}$  as  $\mu_{G,\beta,\lambda}$  conditioned on the desired magnetization.

We use this perspective to show that  $\hat{\mu}_{G,\beta,\eta}$  satisfies  $\ell_\infty$ -independence as follows:

- (1) An extremal combinatorics result on the magnetization of the Ising model from [12] shows that for any  $G \in \mathcal{G}_\Delta$ , the value of  $\lambda$  that achieves expected magnetization  $\eta$  satisfies  $|\log \lambda| > \log \lambda_a$  if  $|\eta| > \eta_a$ . This allows us to use an approach by Chen, Liu, and Vigoda [16] to derive  $O(1)$ - $\ell_\infty$ -independence for the Ising model for all such  $\lambda$  based on our zero-freeness assumption.
- (2) We next show that the probability under  $\mu_{G,\beta,\lambda}$  of drawing a configuration with exactly the correct magnetization is sufficiently large, and that this probability does not change significantly after conditioning on the spin of a vertex. For the former, a lower bound of  $\Theta(1/\sqrt{n})$  can be derived from existing local central limit theorems for the expected number of  $+1$  spins [12]. For the latter, we perform a similar analysis to [33] and use an Edgeworth expansion to prove that conditioning on the spin of a vertex changes this probability by at most  $O(n^{-3/2})$ . For both results it is crucial that the Ising model satisfies sufficiently strong zero-freeness assumptions for all considered  $\lambda$ .

The above discussion indicates how we obtain spectral independence for  $\hat{\mu}_{G,\beta,\eta}$ . The bulk of our work comes from leveraging this to derive a lower bound on the spectral gap of the down-up walk. This requires us to prove that spectral independence also holds when an arbitrary vertex set  $U \subset V$  with  $|U| < k$  is fixed (or *pinned*) to have spin  $+1$ . Such pinning interfere with the proof strategy above for several reasons. First of all, pinning vertices to  $+1$

decreases the  $\lambda$  that we need to choose to obtain the desired magnetization  $\eta$ . In particular, if we aim for  $\eta > \eta_a$ , this might cause the required value of  $\lambda$  to leave the regime in which zero-freeness (and  $\ell_\infty$ -independence) for the Ising model is guaranteed. We circumvent this by observing that the Kawasaki dynamics is symmetric under swapping  $+1$  and  $-1$  spins. Hence, it suffices to consider  $\eta < -\eta_a$ , and an application of the FKG inequality ensures that we only need to consider  $\lambda < 1/\lambda_a(\Delta, \beta)$  for all relevant pinnings.

The second difficulty is that once the number of free vertices  $k - |U|$  becomes sub-linear in  $n$ , both the local central limit theorem and the Edgeworth expansion can fail. Similar to [33], we solve this issue by using the localization framework by Chen and Eldan [14], which allows us to factorize the spectral gap of the down-up walk into the spectral gaps of two Markov chains that are easier to analyze. The first chain is a generalization of the down-up walk that updates  $\Theta(n)$  vertices in each step, and we can analyze its spectral gap based on the spectral independence result described above using the local-to-global framework for local spectral expanders [1, 2, 15, 17]. The second walk is a simple down-up walk but with a set of vertices  $U \subset V$  pinned to  $+1$ . In particular, we need to show that there is some  $\alpha > 0$  (depending on  $\beta$  and  $\Delta$ ) such that for  $k - |U| \leq \alpha n$ , the spectral gap of such a pinned down-up walk is bounded below by  $\Omega(1/n)$ .

For bounding the spectral gap of the pinned walk, we use a coupling argument. Specifically, we construct a suitable metric on the state space such that the distance between two coupled copies of the Markov chain contracts in expectation in each step. For the independent set model studied in [33], such a contracting coupling is well known, appearing in the original “path coupling” paper of Bubley and Dyer [7]. In contrast, for the fixed-magnetization Ising model, no such result exists, and the default choice of coupling (sometimes called the identity coupling) and metric (the number of vertices on which both configurations differ) does not exhibit the desired contraction. Roughly speaking, this is because the ferromagnetism can cause certain types of disagreements to increase the probability that new disagreements are created. We overcome this problem by studying a refined metric, which assigns different weights to “good” and “bad” disagreements in a way that guarantees that distances under this new metric decrease in expectation under the coupling, thus establishing the desired bound on the spectral gap.

### 1.1.2 Slow Mixing

For the slow mixing results, we leverage the connection between the Ising model on the infinite tree  $\mathbb{T}_\Delta$  and the behavior of the model on a uniformly random  $\Delta$ -regular graph. In the relevant range of parameters ( $\beta > \beta_u$ ,  $1 < \lambda < \lambda_u$ ) there are two distinct Ising Gibbs measures on  $\mathbb{T}_\Delta$ , the “plus measure” and the “minus measure.” On the random graph these two Gibbs measures manifest themselves as a dominant and subdominant metastable state: sets of configurations for which the Glauber dynamics take exponential time to escape from. The existence of multiple metastable states immediately shows slow mixing of the Glauber dynamics (Theorem 2), and we then use this to construct a graph on which the Kawasaki dynamics is slow mixing, proving Theorem 1,(2).

To do this, we exhibit the existence of a bottleneck in the state space of the model on a  $\Delta$ -regular graph  $H$  constructed as the disjoint union of several copies of a random  $\Delta$ -regular graph. We define two different subsets of configurations of the fixed-magnetization Ising model on  $H$ : in the set of configurations  $S_1$ , each copy of the random graph comprising  $H$  has magnetization  $\eta$ ; in the set  $S_2$ , some copies have magnetization approximately  $\eta_+ > \eta$  and some copies have magnetization approximately  $\eta_- < \eta$  (chosen in such a way that their average is  $\eta$ ). We then show that a third set  $S_3$  separates  $S_1$  and  $S_2$  (under single-step

updates of the Kawasaki dynamics) and carries exponentially less probability mass in the fixed-magnetization Ising model than either  $S_1$  or  $S_2$ . Via a standard conductance argument this proves exponentially slow mixing of the Kawasaki dynamics.

Bounds on the weights of the sets  $S_1, S_2$ , and  $S_3$  will follow from the existence of the metastable states on the random graph. One metastable state consists of configurations with magnetization close to  $\eta_{\Delta, \beta, \lambda}^+$  and the other consists of configurations with magnetization close to  $\eta_{\Delta, \beta, \lambda}^-$ . That is, the two metastable states are in correspondence with the two distinct extremal Gibbs measures on  $\mathbb{T}_\Delta$  (which is why  $\lambda < \lambda_u$  is crucial).

Identifying the metastable states follows from determining which states (organized according to their magnetizations) contribute significantly to the partition function  $Z_G(\beta, \lambda)$  of the Ising model on the random  $\Delta$ -regular graph. A first guess about how much each state contributes to  $Z_G(\beta, \lambda)$  would be to take the expected contribution. The exponential order of this expectation is captured by a function  $f_{\Delta, \beta, \lambda}(\eta)$ . From [29], we know that the critical points of this function correspond to fixed points of a recursion on  $\mathbb{T}_\Delta$ , and that the second-moment method can be used to lower bound the contribution of the state with magnetization  $\eta$ , where  $\eta$  is the maximum of  $f_{\Delta, \beta, \lambda}(\eta)$ . This suffices to determine the dominant state of the Ising model on the random graph (as was done in much greater generality by Dembo and Montanari in [23]).

To identify subdominant metastable states, however, we need to analyze the contribution of states with magnetization  $\eta$  when  $\eta$  is a local maximum of  $f_{\Delta, \beta, \lambda}(\eta)$ . For this we follow the approach of [19] utilizing non-reconstruction in planted models. While their setting is the  $q$ -state Potts model for  $q \geq 3$ , many of their results can be translated to our context of the external-field Ising model. We discuss their techniques in greater detail in Section 3.2 and in the full paper [36].

When we construct the graph  $H$  as the union of random graphs, we also must understand how the behavior of the fixed-magnetization Ising model relates to that of the Ising model. To do this, we give a new and simple argument in Section 3.2 to bound the probability of hitting a given magnetization in the Ising model.

Interestingly, while the graph on which we show slow mixing is the union of random regular graphs, the behavior of the Kawasaki dynamics on a single copy of the random regular graph can be very different. Recently, Bauerschmidt, Bodineau, and Dagallier [4] (see also [5]) showed that the local Kawasaki dynamics for the fixed-magnetization Ising model mixes in time  $O(n \log^6 n)$  on random  $\Delta$ -regular graphs at all magnetizations when  $\beta < 1/(8\sqrt{\Delta-1})$ . In particular, when  $\Delta$  is sufficiently large this regime of fast mixing includes parameters outside the tree uniqueness phase, i.e. inside the range of parameters for which we prove exponentially-slow mixing in the worst case over graphs in  $\mathcal{G}_\Delta$ .

## 1.2 Outline

In Section 2, we collect preliminary results that will be used in our proofs. In Section 3 we give a more detailed overview on our main steps for proving Theorem 1 and Theorem 2. In particular, in Section 3.1, we discuss our fast-mixing result, Theorem 1,(1), and in Section 3.2 we discuss our slow-mixing results, Theorem 1,(2) and Theorem 2. All proofs and more details can be found in the full version of the paper [36].

## 2 Preliminaries

Throughout the paper and unless otherwise stated, we will make the following assumptions:  $\Delta \geq 3$  is fixed,  $\beta \geq 0$ ,  $G = (V, E) \in \mathcal{G}_\Delta$ , and  $n = |V|$ .

We will often switch between notation of  $\eta$  for the magnetization per vertex and  $k = \lfloor \frac{\eta+1}{2} n \rfloor$  for the number of  $+1$  spins in such a configuration. We will thus abuse notation and write  $\hat{\mu}_{G,\beta,k}$  for  $\hat{\mu}_{G,\beta,\eta}$  and  $\hat{Z}_{G,k}(\beta)$  for  $\hat{Z}_{G,\eta}(\beta)$  when it makes things more clear. We will also on occasion drop  $G$  and  $\beta$  from the subscripts of our Gibbs measure notation as well as the subscripts and argument of our partition function notation when  $G$  and  $\beta$  do not play a role in the proofs.

## 2.1 Ising Model on the Infinite Tree

Let  $\mathbb{T}_\Delta$  denote the infinite  $\Delta$ -regular tree. Since it has infinitely many vertices, one cannot define the Ising model on  $\mathbb{T}_\Delta$  via (1). Instead, the Dobrushin-Lanford-Ruelle equations can be used to define “infinite-volume Gibbs measures” for the Ising model and other spin models on infinite graphs. This approach says that a probability measure  $\mu$  on  $\{\pm 1\}^{V(\mathbb{T}_\Delta)}$  is a Gibbs measure for the Ising model at inverse temperature  $\beta$  and external field  $\lambda$  if the conditional measure on any finite set of vertices given a configuration on the complement is the Ising model defined by (1) with the appropriate boundary conditions. See [30] for more details.

A main question about Gibbs measures on infinite graphs is whether for a given specification of parameters (i.e.  $\beta$  and  $\lambda$  in the Ising case) and a given infinite graph  $G$  there is a unique Gibbs measure or multiple distinct Gibbs measures. The transition between uniqueness and non-uniqueness as a parameter varies marks a phase transition.

Understanding uniqueness and non-uniqueness of the Ising model on  $\mathbb{T}_\Delta$  is relatively simple because of monotonicity and the FKG inequality. There are two extreme infinite-volume Gibbs measures in the sense of maximizing or minimizing the probability that a fixed vertex of  $\mathbb{T}_\Delta$  gets a  $+1$  spin: the “ $+$  measure” on  $\mathbb{T}_\Delta$  is the Gibbs measure realized by taking a weak limit of finite-volume Gibbs measures on depth  $N$  truncations of  $\mathbb{T}_\Delta$  with boundary vertices assigned  $+1$  spins; the “ $-$  measure” is the weak limit of finite-volume measures with boundary vertices receiving  $-1$  spins.

The quantities  $\eta_{\Delta,\beta,\lambda}^+$  and  $\eta_{\Delta,\beta,\lambda}^-$  are the respective expectations of  $\sigma_v$  (for any fixed  $v$  in  $\mathbb{T}_\Delta$ ) under these two Gibbs measures. The quantities can be calculated as solutions to fixed point equations (see e.g. [6]), giving

$$\eta_{\Delta,\beta,\lambda}^+ = \tanh(L^* + \operatorname{artanh}(\tanh(L^*) \tanh(\beta/2)))$$

where  $L^*$  is the largest solution to

$$L = \log(\lambda) + (\Delta - 1)\operatorname{artanh}(\tanh(L) \tanh(\beta/2)).$$

The following proposition summarizes information about  $\eta_{\Delta,\beta,\lambda}^+$ ,  $\eta_{\Delta,\beta,\lambda}^-$  and Gibbs uniqueness that we will use (all follow from the results in [6]).

► **Proposition 5.** *Fix  $\Delta \geq 3$ .*

- *There is uniqueness of Gibbs measure for the Ising model with parameters  $\beta, \lambda$  on  $\mathbb{T}_\Delta$  if and only if  $\eta_{\Delta,\beta,\lambda}^+ = \eta_{\Delta,\beta,\lambda}^-$ .*
- *For  $\beta \leq \beta_u(\Delta) = \ln\left(\frac{\Delta}{\Delta-2}\right)$ , there is uniqueness for all  $\lambda$ .*
- *For  $\beta > \beta_u(\Delta)$  there is  $\lambda_u > 1$  so that there is uniqueness if and only if  $|\log \lambda| > \log \lambda_u$ .*
- *$\eta_{\Delta,\beta,\lambda}^+$  is continuous and strictly increasing in  $\lambda$  on the interval  $[1, \infty)$ . In particular, recall that  $\eta_c(\Delta, \beta) = \eta_{\Delta,\beta,1}^+$  and  $\eta_u(\Delta, \beta) = \eta_{\Delta,\beta,\lambda_u}^+$ ; then for every  $\eta \in [\eta_c, \eta_u]$  there is  $\lambda \in [1, \lambda_u]$  so that  $\eta_{\Delta,\beta,\lambda}^+ = \eta$ .*

Finally, it will be important to bound the expected magnetization in the Ising model for given  $\beta, \lambda$  and any  $G \in \mathcal{G}_\Delta$ . The bound is an extremal result proved in [12].



► **Theorem 6** ([12, Theorem 3]). For  $G \in \mathcal{G}_\Delta$ ,  $\lambda \geq 1$ , and  $\beta \geq 0$ ,

$$\mathbb{E}_{\sigma \sim \mu_{G,\beta,\lambda}}[\eta(\sigma)] \leq \eta_{\Delta,\beta,\lambda}^+.$$

## 2.2 Pinned Models

For the fast-mixing argument, we will frequently consider pinned versions of our models, meaning conditioned on some subset of vertices having been assigned a particular spin. For  $U \subset V$ , we call a function  $\tau_U : U \rightarrow \{+1, -1\}$  a *pinning* on  $U$ . We write  $\Omega^{\tau_U} = \{\sigma \in \Omega \mid \forall u \in U : \sigma(u) = \tau_U(u)\}$  for the set of Ising configurations on  $G$  that agree with  $\tau_U$  on  $U$ . The *Ising partition function with pinning*  $\tau_U$  is defined as

$$Z_G^{\tau_U}(\beta, \lambda) = \sum_{\sigma \in \Omega^{\tau_U}} \lambda^{|\sigma|^+} e^{\beta m_G(\sigma)},$$

and the *Ising model under pinning*  $\tau_U$  is defined by Gibbs measure

$$\mu_{G,\beta,\lambda}^{\tau_U}(\sigma) = \frac{\mathbb{1}_{\sigma \in \Omega^{\tau_U}} \lambda^{|\sigma|^+} e^{\beta m_G(\sigma)}}{Z_G^{\tau_U}(\beta, \lambda)}.$$

Note that for  $\lambda > 0$ , it holds that  $\mu_{\beta,\lambda}^{\tau_U}$  is a well-defined probability distribution with support  $\Omega^{\tau_U}$ . We allow for the case  $U = \emptyset$ , which is equivalent to the unpinned Ising model. Often,  $\tau_U$  will be the constant  $+1$  function on  $U$ , in which case we write  $\Omega^U$ ,  $Z_G^U$  and  $\mu_{\beta,\lambda}^U$ .

Analogously to the Ising model, we will also impose pinnings on the fixed-magnetization model. To this end, set  $\Omega_k^{\tau_U} = \{\sigma \in \Omega_k : \forall u \in U : \sigma(u) = \tau_U(u)\}$  and define the *fixed-magnetization partition function with pinning*  $\tau_U$  as

$$\hat{Z}_{G,k}^{\tau_U}(\beta) = \sum_{\sigma \in \Omega_k^{\tau_U}} e^{\beta m_G(\sigma)}.$$

The *fixed-magnetization Ising model under pinning*  $\tau_U$  is a probability measure with support  $\Omega_k^{\tau_U}$  defined by

$$\hat{\mu}_{G,\beta,k}^{\tau_U}(\sigma) = \frac{\mathbb{1}_{\sigma \in \Omega_k^{\tau_U}} e^{\beta m_G(\sigma)}}{\hat{Z}_{G,k}^{\tau_U}(\beta)}.$$

Throughout the paper, we assume  $|\tau_U|^+ \leq k$  so that the expression above is well-defined. As with the Ising model, we write  $\Omega_k^U$ ,  $\hat{Z}_{G,k}^U$  and  $\hat{\mu}_{G,\beta,k}^U$  when  $\tau_U$  is the constant  $+1$  function.

## 2.3 Kawasaki Dynamics, Down-up Walk, and Glauber Dynamics

Here we formally define the three Markov chains that we will analyze. Our main object of study is the Kawasaki dynamics for the fixed-magnetization Ising model. For this, we fix a size  $k$  where  $1 \leq k \leq |V| - 1$ .

► **Definition 7** (Kawasaki dynamics). The *Kawasaki dynamics* on  $\Omega_k$  is a Markov chain  $\mathcal{K}_{\beta,k} = (X_t)_{t \geq 0}$  given by the following update rule:

1. Pick  $u \in X_t^{-1}(+1)$  and  $w \in X_t^{-1}(-1)$  uniformly at random, and set  $X \in \Omega_k$  such that  $X(v) = X_t(v)$ ,  $X(w) = X_t(w)$ , and  $X(u) = X_t(u)$  for  $u \neq v, w$ .
2. Set  $X_{t+1} = X$  with probability  $\min \left\{ 1, \frac{\hat{\mu}_{G,\beta,k}(X)}{\hat{\mu}_{G,\beta,k}(X_t)} \right\}$ , and set  $X_{t+1} = X_t$  otherwise.

## 56:12 Fast and Slow Mixing of the Kawasaki Dynamics

In other words, the Kawasaki dynamics chooses two vertices with opposite spins and swaps their spins with probability proportional to the change in monochromatic edges.

For proving fast mixing of the Kawasaki dynamics, we use the down-up walk on the  $+1$  spins as a proxy for our analysis. Here we will also need to consider the Markov chain under plus pinnings.

► **Definition 8** (Down-up walk with plus pinnings). For  $U \subset V$  and with  $|U| < k$  we define the  $+1$ -down-up walk on  $\Omega_k^U$  as a Markov chain  $\mathcal{P}_{\beta,k}^U = (Y_t)_{t \geq 0}$ , given by the following update rule:

1. Pick  $v \in Y_t^{-1}(+1) \setminus U$  uniformly at random and set  $W = Y_t^{-1}(+1) \setminus \{v\}$ .

2. Draw  $Y_{t+1}$  from  $\hat{\mu}_{G,\beta,k}^W$ .

We write  $\mathcal{P}_{\beta,k}$  if  $U = \emptyset$ .

The following observation is easy to check.

► **Observation 9.**  $\mathcal{K}_{\beta,k}$  and  $\mathcal{P}_{\beta,k}$  are ergodic and reversible with respect to  $\hat{\mu}_{\beta,k}$ . Moreover, there is a constant  $C \geq 1$  that only depends on  $\Delta$  and  $\beta$  such that for all  $\sigma_1 \neq \sigma_2$

$$\frac{1}{C} \cdot \mathcal{P}_{\beta,k}(\sigma_1, \sigma_2) \leq \mathcal{K}_{\beta,k}(\sigma_1, \sigma_2) \leq C \cdot \mathcal{P}_{\beta,k}(\sigma_1, \sigma_2).$$

Lastly, we also consider the Glauber dynamics for the Ising model.

► **Definition 10** (Glauber dynamics). The Glauber dynamics on  $\Omega$  is a Markov chain  $(X_t)_{t \geq 0}$ , given by the following update rule:

1. Pick  $v \in V(G)$  uniformly at random.

2. For  $u \neq v$ , set  $X_{t+1}(u) = X_t(u)$ , and sample  $X_{t+1}(v)$  from the marginal distribution at  $v$  conditioned on  $X_{t+1}(N(v))$ .

## 2.4 Mixing Times

Our goal in analyzing the Kawasaki dynamics is to understand the *mixing time* of this Markov chain. Given two probability distributions  $\mu$  and  $\nu$  on probability space  $(\Omega, \mathcal{A})$ , let

$$\|\mu - \nu\|_{\text{TV}} := \sup_{A \in \mathcal{A}} |\mu(A) - \nu(A)|$$

be the *total variation distance* between  $\mu$  and  $\nu$ . For a Markov chain on  $\Omega$  with transition matrix  $P$  and unique stationary distribution  $\pi$ , we may then define

$$d(t) := \max_{x \in \Omega} \|P^t(x, \cdot) - \pi\|_{\text{TV}}.$$

► **Definition 11.** The mixing time is

$$\tau_{\text{mix}} = \inf \left\{ t : d(t) \leq \frac{1}{4} \right\}.$$

See, e.g., [37] for background on Markov chains and mixing times. We use several different techniques to analyze the mixing time of the Kawasaki dynamics, which we now describe.

### 2.4.1 Upper Bounds on Mixing Time

A common way to upper-bound the mixing time of a reversible Markov chain  $P$  is by lower-bounding its spectral gap, which can be defined via the following variational characterization.

► **Definition 12.** Let  $P$  be a transition matrix that is reversible with respect to  $\pi$ . We denote by  $\mathbf{gap}(P)$  the spectral gap (or Poincaré constant) of  $P$ , which is defined as the largest constant  $\gamma$  such that  $\gamma \text{Var}_\pi(f) \leq \mathcal{E}_P(f, f)$  for any function  $f : \Omega \rightarrow \mathbb{R}$ , where  $\text{Var}_\pi(f)$  is the variance of  $f$  with respect to  $\pi$  and  $\mathcal{E}_P$  is the Dirichlet form of  $P$ , given by

$$\mathcal{E}_P(f, g) = \frac{1}{2} \sum_{x, y \in \Omega} (f(x) - f(y))(g(x) - g(y))P(x, y)\pi(x) \quad f, g : \Omega \rightarrow \mathbb{R}.$$

Using this characterization of the spectral gap, we have the following observation.

► **Observation 13.** Suppose  $P_1$  and  $P_2$  are transition matrices that are both reversible with respect to  $\pi$ . If there are constants  $\alpha_1, \alpha_2 > 0$  such that  $\alpha_1 \cdot P_1(x, y) \leq P_2(x, y) \leq \alpha_2 \cdot P_1(x, y)$  for all  $x \neq y$ , then  $\alpha_1 \cdot \mathbf{gap}(P_1) \leq \mathbf{gap}(P_2) \leq \alpha_2 \cdot \mathbf{gap}(P_1)$ .

On account of Observation 9, this allows us to study the spectral gap of the down-up walk  $\mathcal{P}_{\beta, k}$  instead of the Kawasaki dynamics  $\mathcal{K}_{\beta, k}$ .

An upper bound on the mixing time of an ergodic, reversible Markov chain with transition matrix  $P$  can be obtained from its spectral gap via the following standard relationship (see [37, Theorem 12.4]):

$$\tau_{\text{mix}} \leq \mathbf{gap}(P)^{-1} \cdot \log \left( \frac{4}{\min_{x \in \Omega} \pi(x)} \right).$$

There are various ways to obtain bounds on the spectral gap of a Markov chain, one of which is to construct a contracting coupling. For a transition matrix  $P$ , we say that a Markov chain  $(X_t, Y_t)_{t \geq 0}$  on  $\Omega \times \Omega$  is a *coupling* of  $P$  with itself if each of the marginal processes  $(X_t)_{t \geq 0}$  and  $(Y_t)_{t \geq 0}$  is a Markov chain with transition matrix  $P$ . We use this notion to bound the spectral gap.

► **Theorem 14** ([37, Theorem 13.1]). Suppose  $\Omega$  is finite and let  $(X_t, Y_t)_{t \geq 0}$  be a coupling of  $P$  with itself. If there is a constant  $c > 0$  and a function  $\rho : \Omega \times \Omega \rightarrow \mathbb{R}_{\geq 0}$  such that  $\rho(x, y) = 0$  if and only if  $x = y$ , and for all  $t \in \mathbb{Z}_{\geq 0}$  it holds that

$$\mathbb{E}[\rho(X_{t+1}, Y_{t+1}) \mid X_t, Y_t] \leq (1 - c)\rho(X_t, Y_t),$$

then the spectral gap of  $P$  is at least  $c$ .

We will use Theorem 14 to show that the down-up walk  $\mathcal{P}_{\beta, k}^U$  has a spectral gap of  $\Omega(1/k)$  whenever  $k - |U| \leq \alpha n$  for some  $\alpha$  depending on  $\Delta$  and  $\beta$ . In particular, by the symmetry of the Kawasaki dynamics under swapping all spins, this proves a spectral gap of  $\Omega(1/k)$  for  $\mathcal{K}_{\beta, k}$  if  $k \leq \alpha n$  or  $k \geq (1 - \alpha)n$ , but it does not cover the full regime of Theorem 1, (1).

To prove the full result of Theorem 1, (1), we prove that  $\hat{\mu}_{\beta, k}^U$  satisfies spectral independence for suitable  $k \in \mathbb{N}$  and sets  $U \subset V$ . Spectral independence is a property of the stationary distribution  $\pi$  of a Markov chain, and it was recently used to bound the spectral gap and prove rapid mixing of various chains [1, 2, 14, 15, 17, 33]. For the following discussion of spectral independence, we restrict ourselves to distributions on  $\Omega = 2^V$  where  $V$  is some finite set (e.g., the vertices of a graph). Note that this encompasses both the fixed-magnetization Ising

## 56:14 Fast and Slow Mixing of the Kawasaki Dynamics

model as well as the Ising model, by associating  $S \in \Omega$  with the Ising configuration that maps all vertices in  $S$  to  $+1$ . In this setting, we adopt the following notation: for a distribution  $\pi$  on  $\Omega$ , a subset  $S$  drawn from  $\pi$ , and  $v \in V$ , we write  $\pi(v)$  for the probability that  $v \in S$  and  $\pi(\bar{v})$  for the probability that  $v \notin S$ . We extend this to conditional probabilities, writing for example  $\pi(v | \bar{u})$  for the probability that  $v \in S$  given  $u \notin S$ .

► **Definition 15.** *The influence matrix of a distribution  $\pi$  on  $2^V$  is the matrix  $M_\pi \in \mathbb{R}^{n \times n}$  with entries*

$$M_\pi[u, v] = \begin{cases} 0 & \text{if } \pi(u) = 0 \\ \pi(v | u) - \pi(v) & \text{otherwise} \end{cases}$$

Using this definition of  $M_\pi$ , the definition of spectral independence of  $\pi$  is as follows.

► **Definition 16.** *A probability distribution  $\pi$  on  $2^V$  is called  $C$ -spectrally independent (for  $C \geq 0$ ) if the largest eigenvalue of  $M_\pi$  is at most  $C$ .*

Since directly bounding the largest eigenvalue of  $M_\pi$  is usually challenging, a common approach is to bound the  $\ell_\infty$ -norm of  $M_\pi$  instead. This leads to the stronger notion of  $\ell_\infty$ -independence.

► **Definition 17.** *A probability distribution  $\pi$  on  $2^V$  is  $C$ - $\ell_\infty$ -independent (for  $C \geq 0$ ) if*

$$\|M_\pi\|_\infty := \max_{u \in V} \sum_{v \in V} |M_\pi[u, v]|$$

*is at most  $C$ .*

► **Remark 18.** There are various definitions of the pairwise influence matrix in the literature [2, 15, 17]. For spin systems with two possible states for each vertex (such as the Ising model), pairwise influence is commonly defined as  $M_\pi[u, v] = \pi(v | u) - \pi(v | \bar{u})$ . However, note that switching between the two definitions only changes the spectral radius by some constant factor, provided that  $\pi(v)$  is uniformly bounded away from 0 and 1. Since this is the case for the Ising model, given that  $\lambda > 0$ , existing spectral independence results such as [17] carry over to our definition. Moreover, Definition 15 is arguably more natural for canonical ensembles, such as the fixed-magnetization Ising model, as it relates more directly to local spectral expansion of the associated simplicial complex (see [36] for details).

There are different ways to derive bounds on the spectral gap of a Markov chain from spectral independence. The most popular approach is the use of *local-to-global theorems*, which are applicable whenever the Markov chain in question can be represented as a down-up walk on a suitable weighted simplicial complex [1, 2, 15, 17]. Local-to-global theorems allow us to express the spectral gap of the down-up walk in terms of spectral gaps of local walks on the complex, which can then be related to the spectrum of the pairwise influence matrix.

A more recent framework was introduced by Chen and Eldan [14] and uses *localization schemes*. A localization scheme maps a probability distribution  $\pi$  on  $\Omega$  to a localization process – a random sequence of probability measures that interpolates between  $\pi$  and a random Dirac measure. Via the localization process, a localization scheme gives rise to a Markov chain with stationary distribution  $\pi$ . Provided that the localization process exhibits a property called “approximate conservation of variance,” this can be used to bound the spectral gap of the associated Markov chain. For a broad class of localization schemes, approximate conservation of variance follows if all measures along the localization process exhibit spectral independence. Since we are studying the fixed-magnetization Ising model, we

are particularly concerned with distributions  $\pi$  on  $\Omega_k$ . In this setting, the canonical choice for a localization scheme is the subset simplicial-complex localization (see [14, Example 5]), and the natural associated Markov chain is the down-up walk  $\mathcal{P}_{\beta,k}$ .

The main difficulty in applying the above frameworks in our setting is that they usually assume  $O(1)$ -spectral independence of the pinned distributions  $\hat{\mu}_{\beta,k}^U$  for all  $U \subset V$  with  $0 \leq |U| \leq k - 1$ . Unfortunately, we will not be able to derive spectral independence for all such  $U$ . Moreover, for the localization framework, it is not clear if the subset simplicial-complex localization allows us to derive approximate conservation of variance from spectral independence. To overcome these difficulties, we use an argument similar to that of Jain, Michelen, Pham and Vuong [33]. We combine the techniques above as follows: first, we use a localization scheme to show that for any  $\ell \leq k - 1$ , the spectral gap of  $\mathcal{P}_{\beta,k}$  is bounded below by the product of the spectral gap of the pinned down-up walk  $\mathcal{P}_{\beta,k}^U$  for any  $U \subset V$  with  $|U| = \ell$  and the spectral gap of the  $(k, \ell)$ -down-up walk, a modified version of  $\mathcal{P}_{\beta,k}$  that resamples  $k - \ell$  plus spins in each step. Choosing  $\ell$  such that  $k - \ell \leq \alpha n$  for some suitable  $\alpha > 0$ , we can use a coupling argument as discussed before to show that  $\text{gap}(\mathcal{P}_{\beta,k}^U) \in \Omega(1/k)$  for every  $U \subset V$  with  $|U| = \ell$ . To lower-bound the spectral gap of the  $(k, \ell)$ -down-up walk, we use a local-to-global theorem by Chen, Liu and Vigoda [15]. This only requires us to show that  $\hat{\mu}_{\beta,k}^W$  satisfies  $O(1)$ -spectral independence for all  $W \subset V$  with  $k - |W| \geq \alpha' n$  for some  $0 < \alpha' < \alpha$ . The range of  $k$  for which we can show this  $O(1)$ -spectral independence leads to the magnetization range given in Theorem 1,(1).

### 2.4.2 Lower Bounds on Mixing Time

To prove slow mixing, we exhibit the existence of a bottleneck in the state space, a set of configurations which separates two parts of the state space and carries an exponentially smaller probability in the stationary distribution than either of the two parts. The following lemma captures a simple form of this argument, often phrased in terms of *conductance*, for proving lower bounds on the mixing times of Markov chains.

► **Lemma 19.** *Let  $(X_t)_{t \geq 0}$  be a Markov chain on the state space  $\Omega$  with stationary distribution  $\pi$ . Suppose there exists disjoint sets  $S_1, S_2, S_3 \subset \Omega$  so that the following hold:*

- *For the chain to pass from  $S_2$  to  $S_1$  it must pass through  $S_3$ ;*
- $\pi(S_1) \geq \pi(S_2)$
- $\pi(S_3) \leq e^{-\Omega(n)} \pi(S_2)$ .

*Then the mixing time of the chain  $(X_t)$  is  $\exp(\Omega(n))$ .*

The statement is an immediate corollary of, e.g., [25, Claim 2.3].

To prove Theorem 2, we define  $S_1, S_2, S_3$  to be sets of configurations with certain magnetizations.  $S_1$  will be those configurations whose magnetization per vertex is close to that of the plus measure on  $\mathbb{T}_\Delta$  (when  $\lambda > 1$ );  $S_2$  will be those whose magnetization per vertex is close to that of the minus measure; and  $S_3$  will be configurations whose magnetization is just larger than that of  $S_2$ .

To prove Theorem 1,(2), we consider a graph  $H$  made up of disjoint copies of a random regular graph. We define  $S_1$  to be the set of configurations with magnetization  $\eta$  on each copy;  $S_2$  will be a set of configurations with magnetization  $\eta_+$  on some copies and  $\eta_-$  on others, for  $\eta_- < \eta < \eta_+$ , such that the overall magnetization is  $\eta$ . Again  $S_3$  will be a neighborhood of  $S_2$ . In both cases, the main work will be in verifying the conditions of Lemma 19.

### 2.5 Thresholds for Zero-Freeness and Spectral Independence

The definition of  $\lambda_a(\Delta, \beta)$  is based on viewing the Ising partition function as a polynomial in the (complex) variable  $\lambda$ . We write  $\mathcal{N}(z, \delta)$  for the open ball of radius  $\delta > 0$  around  $z \in \mathbb{C}$ .

► **Definition 20** (Absolute zero-freeness). *Given  $\beta \geq 0$ ,  $\Delta \in \mathbb{N}$ ,  $\lambda > 0$  and  $\delta > 0$ , we say that the Ising model is absolutely  $\delta$ -zero-free at activity  $\lambda$  if for all graphs  $G \in \mathcal{G}_\Delta$ , all pinnings  $\tau_U$  with  $U \subseteq V$  and all  $\lambda' \in \mathcal{N}(\lambda, \delta)$  it holds that  $Z_G^{\tau_U}(\beta, \lambda') \neq 0$ .*

We now define  $\lambda_a(\Delta, \beta)$  as follows.

► **Definition 21.** *For  $\Delta \in \mathbb{N}$  and  $\beta \geq \beta_u(\Delta)$  we set  $\lambda_a(\Delta, \beta)$  to be the smallest  $\lambda_a \geq 1$  such that for every compact set  $D \subset (\lambda_a, \infty)$  there is some  $\delta > 0$  such that for all  $\lambda \in D$  the Ising model is absolutely  $\delta$ -zero-free at  $\lambda$ .*

An important implication of absolute zero-freeness is given in the following theorem. Its proof follows a similar argument to those in [16] while using the ferromagnetism of the model and Montel’s theorem (see [49]) to avoid the requirement of multivariate zero-freeness. The proof can be found in the full version of the paper [36].

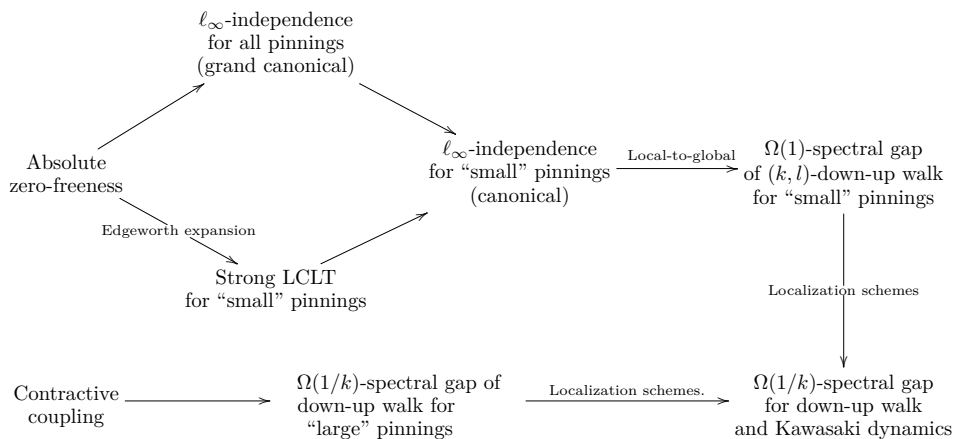
► **Theorem 22.** *Fix  $\beta \geq 0$  and  $\Delta \in \mathbb{N}$ . Let  $D \subset \mathbb{R}_{>0}$  be compact and assume there is some  $\delta > 0$  such that the ferromagnetic Ising model is absolutely  $\delta$ -zero-free at every  $\lambda \in D$ . Then, there is some constant  $C > 0$ , only depending on  $D$ ,  $\lambda$ ,  $\beta$  and  $\Delta$ , such that for all  $\lambda \in D$ ,  $G \in \mathcal{G}_\Delta$  and all pinnings  $\tau_U$  it holds that  $\hat{\mu}_{G, \beta, \lambda}^{\tau_U}$  is  $C$ - $\ell_\infty$ -independent.*

## 3 Main Statements and Proof Structure

We briefly state the most important steps for showing Theorem 1. All proofs and intermediate steps are omitted and can be found in the full version of the paper [36].

### 3.1 Rapid Mixing

We start with discussing our proof of the rapid mixing result in part (1) of Theorem 1. The structure of the entire proof is illustrated in Figure 3.



■ **Figure 3** The structure of the rapid mixing proof.

All results in this subsection are given in the context of the following assumptions.

► **Condition 23.**

1. Let  $\beta \geq 0$ , and let  $D \subset \mathbb{R}_{>0}$  be compact such that there is some  $\delta > 0$  for which the Ising model is absolutely  $\delta$ -zero-free for all  $\lambda \in D$ . Further, let  $\lambda \in D$ .
2. Let  $\alpha \in [0, 1)$ , let  $U \subset V$  with  $|U| \leq \alpha n$  and let  $\tau_U$  be a pinning of  $U$ .
3. Let  $\sigma \sim \mu_{\beta, \lambda}^{\tau_U}$  and let  $X = |\sigma|^+$ .

Our first step is to show that zero-freeness implies a strengthened version of a local central limit theorem for  $X$  via Edgeworth expansion. Using similar arguments as Jain, Michelen, Pham and Vuong [33] for the hard-core model, we obtain the following result.

► **Theorem 24.** *Suppose Condition 23 holds. Let  $d \in \mathbb{N}$  and let  $\ell \in \mathbb{R}$  such that  $\mathbb{E}[X] + \ell \in \mathbb{Z}_{\geq 0}$ . Set  $s = \sqrt{\text{Var}(X)}$  and  $\beta_j = \frac{\kappa_j(X)}{j!s^j}$  for all  $j \in \mathbb{N}$ , and write  $H_k(\cdot)$  for the  $k^{\text{th}}$  Hermite polynomial. It holds that*

$$\mu_{\beta, \lambda}^{\tau_U}(X - \mathbb{E}[X] = \ell) = \frac{e^{-\frac{\ell^2}{2s^2}}}{\sqrt{2\pi s}} \left( 1 + \sum_{r \geq 3} H_r(\ell/s) \sum_{k_3, \dots, k_{2d+1}} \prod_{j=3}^{2d+1} \frac{\beta_j^{k_j}}{k_j!} \right) + O(n^{-d})$$

where the inner sum is over tuples  $k_3, \dots, k_{2d+1} \in \mathbb{Z}_{\geq 0}$  such that  $\sum_j k_j \cdot j = r$  and  $\sum_j k_j \cdot \frac{j-2}{2} \leq d$ , and the implied constants depend only on  $\Delta, \beta, \delta, D, d$  and  $\alpha$ .

Our next ingredient is to use zero-freeness to obtain a stability result for the cumulants of  $X$  under adding vertices to the pinning. Writing  $\kappa_j(X)$  for the  $j$ th cumulant of  $X$ , we have the following statement.

► **Lemma 25.** *Suppose Condition 23 holds. Let  $v \in V \setminus U$ , and let  $\tau_U, +_v$  denote the pinning on  $U \cup \{v\}$  that maps  $v$  to  $+1$  and all other vertices  $u \in U$  to  $\tau_U(u)$ . Let  $X^+ = |\sigma'|^+$  for  $\sigma' \sim \mu_{\beta, \lambda}^{\tau_U, +_v}$ . For all  $j \in \mathbb{N}$  it holds that  $|\kappa_j(X^+) - \kappa_j(X)| = O(1)$  with implied constants only depending on  $\Delta, \beta, \delta, D$  and  $j$ .*

The analog of Lemma 25 for the hard-core model was proven in [33]. However, their arguments are tailored to the hard-core model and do not apply in our setting. Instead, we provide a more general argument based on an application of Montel’s theorem that is inspired by [43].

Using Theorem 24 and Lemma 25, we get the following stability result for the probability of having exactly  $k$  vertices assigned to  $+1$ .

► **Lemma 26.** *Suppose Condition 23 holds and assume further that  $|U| + 1 \leq \alpha n$ . Let  $k \in \mathbb{Z}_{\geq 0}$  be such that  $|\mathbb{E}[X] - k| \leq L$  for some  $L \in \mathbb{R}_{\geq 0}$ . For all  $v \in V \setminus U$  it holds that*

$$\mu_{\beta, \lambda}^{\tau_U}(X = k) = \Theta(n^{-1/2}), \tag{3}$$

$$\left| \mu_{\beta, \lambda}^{\tau_U}(X = k) - \mu_{\beta, \lambda}^{\tau_U}(X = k \mid \sigma(v) = +1) \right| = O(n^{-3/2}) \tag{4}$$

with implied constants depending only on  $\Delta, \beta, \delta, D, L$  and  $\alpha$ .

Next, recall that by Theorem 22 zero-freeness implies  $\ell_\infty$ -independence for the ferromagnetic Ising model. Combining this with Lemma 26 for a suitable  $\lambda$ , we get the following  $\ell_\infty$ -independence result for the fixed magnetization model.

► **Theorem 27.** *Assume  $0 \leq \beta < \beta_u(\Delta)$  and  $\gamma \in (0, 1/2]$ , or  $\beta \geq \beta_u(\Delta)$  and  $\gamma \in (0, \frac{1-\eta_a}{2})$  for  $\eta_a = \eta_a(\Delta, \beta)$ . For all  $k := \gamma n \in \mathbb{N}$ , all  $\alpha \in [0, \gamma)$  and  $U \subset V$  with  $|U| \leq \alpha n$  it holds that  $\hat{\mu}_{\beta, k}^U$  is  $C$ - $\ell_\infty$ -independent for a constant  $C$  depending only on  $\Delta, \beta, \gamma$  and  $\alpha$ .*

Using Theorem 27, we can apply a local-to-global theorem from [15] to show that for every  $k - \ell \in \Theta(n)$  the spectral gap of the  $(k, \ell)$ -down-up walk is in  $\Omega(1)$ . However, to get the desired spectral gap for  $\mathcal{P}_{\beta,k}$  (and  $\mathcal{K}_{\beta,k}$ ), we require one last ingredient, which is to show that the spectral gap of the pinned down-up walk  $\mathcal{P}_{\beta,k}^U$  is in  $\Omega(1/n)$  whenever  $k = \gamma n$  and  $U \subset V$  are such that  $k - |U|$  is small enough.

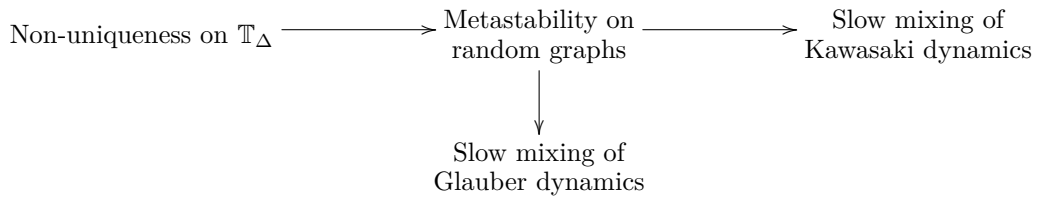
In the setting of fixed-size independent sets studied in [33], such a result was previously known due to Theorem 14 and a path coupling by Buble and Dyer [7]. In contrast, a straightforward application of path coupling with the Hamming metric does not work in our setting. Instead, we introduce a modified metric on the state space that takes into account how likely a disagreement is to spread, which allows us to prove the following result.

► **Lemma 28.** *Let  $G \in \mathcal{G}_\Delta$  with  $n$  sufficiently large. There exists some  $\alpha = \alpha(\Delta, \beta) > 0$  such that for all  $0 < k \leq n/2$  and all  $U \subset V$  with  $0 < k - |U| \leq \alpha n$  it holds that  $\mathcal{P}_{\beta,k}^U$  has spectral gap  $\Omega(1/k)$  with constants depending on  $\beta$  and  $\Delta$ .*

We can now proceed to sketch our proof of the rapid mixing part of Theorem 1. We first note that the Kawasaki dynamics Markov chain is invariant under swapping all spins (i.e. the mapping  $\sigma \mapsto -\sigma$ ), allowing us to focus on  $k \leq n/2$  (or equivalently the magnetization regime  $\eta \leq 0$ ). Moreover, by Observation 9 it suffices to prove the desired spectral gap for the down-up walk  $\mathcal{P}_{\beta,k}$  for the respective values of  $k$ . Using a localization scheme, we argue that the spectral gap of  $\mathcal{P}_{\beta,k}$  is bounded below by the product of  $\inf_{U \in \binom{V}{\ell}} \text{gap}(\mathcal{P}_{\beta,k}^U)$  and the spectral gap of the  $(k, \ell)$ -down-up walk. By Lemma 28, we know that  $\inf_{U \in \binom{V}{\ell}} \text{gap}(\mathcal{P}_{\beta,k}^U) \in \Omega(1/k)$  whenever  $\ell$  is such that  $k - \ell \leq \alpha n$  for some  $\alpha = \alpha(\Delta, \beta) > 0$ . Moreover, by Theorem 27 and a local-to-global theorem from [15], we can derive a  $\Omega(1)$  spectral gap for the  $(k, \ell)$ -down-up walk. Combining both concludes our rapid mixing proof.

### 3.2 Metastability and Slow Mixing

In this section we prove slow-mixing results for both the Ising Glauber dynamics and fixed magnetization Kawasaki dynamics when  $\beta > \beta_u(\Delta)$  and  $|\log \lambda| < \log \lambda_u$  and  $|\eta| < \eta_u$  respectively. The structure of the proof is illustrated below in Figure 4.



■ **Figure 4** The structure of the slow mixing proof.

**Note.** As in the previous section, both perspectives of fixed magnetization per vertex  $\eta$  and fixed size  $k$  will be useful in our arguments. We will use  $Z_{G,\eta}(\beta, \lambda)$  (where we sometimes drop the parameters  $\beta$  and  $\lambda$  for convenience) to denote the contribution to the Ising model partition function  $Z_G(\beta, \lambda)$  from configurations of magnetization  $\eta$ . The notation  $Z_{G,k}(\beta, \lambda)$  will mean the contributions to  $Z_G(\beta, \lambda)$  from configurations of size  $k$ . When  $k = \lfloor n^{\frac{\eta+1}{2}} \rfloor$ , we have  $Z_{G,\eta} = Z_{G,k}$  and will use the notations interchangeably.



Our goal is to understand how configurations of different magnetizations typically contribute to the partition function  $Z_G(\beta, \lambda)$  when  $G$  is a random  $\Delta$ -regular graph. To start, we shift to a slightly different model called the *configuration model*, which we will denote  $\mathbf{G}$ . To generate a graph from this model for a given  $\Delta$  and  $n$ , take  $\Delta$  copies of  $[n]$  and a uniformly random perfect matching on the  $\Delta n$  vertices, and then identify the copies corresponding to the same vertex. This gives a random  $\Delta$ -regular multigraph, and it is well-known that properties holding with high probability for the configuration model also hold with high probability for the uniform random  $\Delta$ -regular graph when  $\Delta$  is constant [34].

We say the model has multiple *metastable states* if the function  $\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E} \log Z_{G,\eta}(\beta, \lambda)$  has more than one local maximum as  $\eta$  varies. A first attempt at understanding this phenomenon would be to look at the first moment, and understand the local maxima of

$$f_{\Delta,\beta,\lambda}(\eta) := \lim_{n \rightarrow \infty} \frac{1}{n} \log \mathbb{E} Z_{G,\eta}(\beta, \lambda) \tag{5}$$

as a function of  $\eta$  (with the crucial distinction between the two functions being the interchange of the expectation and logarithm).

Using computations similar to those found in [18, 19, 29], we can derive an expression for  $f_{\Delta,\beta,\lambda}(\eta)$ . We then proceed by studying the the maxima of  $f_{\Delta,\beta,\lambda}(\eta)$  as a one-variable function with respect to  $\eta$ . By a result in [29] (following [27, 41]), we know that the critical points of  $f_{\Delta,\beta,\lambda}(\eta)$  correspond exactly to fixed points of the *tree recursion* for the Ising model on  $\mathbb{T}_\Delta$ , which are the solutions to the equation

$$R = \frac{\lambda(Re^\beta + 1)^{\Delta-1}}{(R + e^\beta)^{\Delta-1}}. \tag{6}$$

► **Theorem 29** ([29, Theorem 9, Lemma 11]). *There is a 1-to-1 correspondence between the fixed points of the tree recursion given in (6) and the critical points of  $f_{\Delta,\beta,\lambda}(\eta)$ . Moreover, the stable fixed points of the tree recursion given in (6) are in 1-to-1 correspondence with Hessian local maxima of  $f_{\Delta,\beta,\lambda}(\eta)$ .*

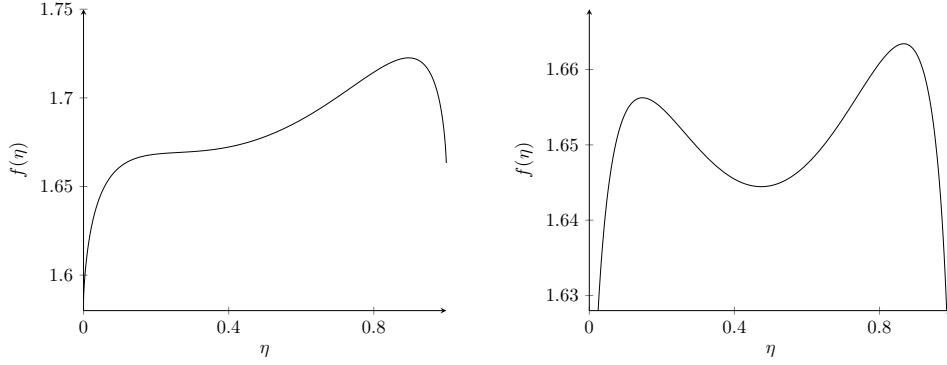
Recall that a fixed point is *stable* if the absolute value of the derivative at that point is less than 1. A local maximum is a *Hessian local maximum* if the Hessian is negative definite at that point. In particular, as our functions are univariate (after fixing  $\Delta, \beta, \lambda$ ), this is simply saying that the second derivative is negative which implies the existence of a local maximum.

For the above theorem to be useful, we need to understand the solutions of (6).

► **Proposition 30.** *For  $\beta > \beta_u$ , the following hold:*

- (1) *If  $|\log \lambda| > \log \lambda_u$ , then (6) has a unique fixed point. It is stable and hence corresponds to the global maximizer of  $f_{\Delta,\beta,\lambda}$ . This maximizer is  $\eta_{\Delta,\beta,\lambda}^+ = \eta_{\Delta,\beta,\lambda}^-$ .*
- (2) *If  $|\log \lambda| = \log \lambda_u$ , then (6) has two distinct fixed points, one of which is stable and corresponds to the global maximizer of  $f_{\Delta,\beta,\lambda}$ . The other corresponds to an inflection point of  $f_{\Delta,\beta,\lambda}$ .*
- (3) *If  $|\log \lambda| < \log \lambda_u$ , then (6) has three distinct fixed points. The largest and the smallest are both stable, corresponding to the only two local maxima of  $f_{\Delta,\beta,\lambda}$ . When  $\lambda > 1$ ,  $\eta_{\Delta,\beta,\lambda}^+$  is the unique global maximizer; when  $\lambda < 1$ ,  $\eta_{\Delta,\beta,\lambda}^-$  is the unique global maximizer; when  $\lambda = 1$  then  $\eta_{\Delta,\beta,\lambda}^+, \eta_{\Delta,\beta,\lambda}^-$  are both global maximizers.*

Portions of this statement have been shown in, for example, [29, 30, 32], and we give a complete proof in [36]. An illustration of  $f_{\Delta,\beta,\lambda}(\eta)$  is given in Figure 5; the left plot appears for  $\lambda > \lambda_u$  (Case 1 above) and the right plot appears for  $1 < \lambda < \lambda_u$  (Case 3 above).



■ **Figure 5** Sketch of the function  $f_{\Delta,\beta,\lambda}(\eta)$  for  $\Delta = 4$ ,  $\beta = \ln(2) + 0.1$ , and (left)  $\lambda = 1.08$ , (right)  $\lambda = 1.01$ .

While the behavior described in part (3) suggests metastability, Proposition 30 is only about the expected partition function, and we will need to show that multiple local maxima exist with high probability over the random graph. This will involve showing a lower bound on the partition function at the two local maxima and an upper bound everywhere else.

Via Markov's inequality, the next statement gives a high probability approximate upper bound on  $Z_{G,\eta}(\lambda)$ .

► **Lemma 31.** *Fix  $\beta \geq 0, \lambda > 0$ . With probability  $1 - o(1)$  over the random  $\Delta$ -regular graph  $G$  on  $n$  vertices, it holds for every  $\eta$  that*

$$Z_{G,\eta}(\lambda) \leq n^2 \cdot \mathbb{E}Z_{G,\eta}(\lambda).$$

We further prove lower bounds on  $Z_{G,\eta}$  for values of  $\eta$  which are local maxima. For a global maximum, this was proved in [29] via the second moment method.

► **Theorem 32** ([29, Theorem 8]). *Fix  $\lambda > 0$  and suppose that  $\eta$  is a global maximizer of  $f_{\Delta,\beta,\lambda}$ . With probability  $1 - o(1)$  over the random  $\Delta$ -regular graph  $G$  on  $n$  vertices,*

$$Z_{G,\eta}(\lambda) \geq \frac{1}{n} \mathbb{E}Z_{G,\eta}(\lambda).$$

We prove the following corresponding statement for the local maximizers.

► **Proposition 33.** *Fix  $\lambda > 0$  and suppose that  $\eta$  is a local maximizer of  $f_{\Delta,\beta,\lambda}$ . For any  $\zeta > 0$ , with probability  $1 - o(1)$  over the random  $\Delta$ -regular graph  $G$  on  $n$  vertices,*

$$Z_{G,\eta}(\lambda) \geq e^{-\zeta n} \mathbb{E}[Z_{G,\eta}(\lambda)].$$

The proof of Proposition 33 follows the template of Coja-Oghlan, Galanis, Goldberg, Ravelomanana, Štefankovič, and Vigoda [19] in proving metastability in the zero-field ferromagnetic Potts model (which in turn used ideas from [3,21]). The argument involves various techniques such as studying the *planted model*, *Nishimori identities* [20], and *non-reconstruction of broadcasting processes* [19,29,39], and it is presented in the full paper [36]. We can now sketch the proofs of our slow mixing results.

### Slow mixing of Glauber Dynamics

We start with sketching our proof of Theorem 2. Let  $\beta > \beta_u(\Delta)$ ,  $\lambda \in [1, \lambda_u)$ , and  $G \sim \mathbf{G}$ . Let  $\eta = \eta_{\Delta,\beta,\lambda}^+$ , the mean magnetization of the root of  $\mathbb{T}_\Delta$  under the  $+$  boundary conditions with external field  $\lambda$ , and let  $\eta_- = \eta_{\Delta,\beta,\lambda}^-$ , the same but under the  $-$  boundary conditions.

As  $\eta$  and  $\eta_-$  are global and local maximizers of  $f_{\Delta,\beta,\lambda}$ , there are  $\epsilon > 0$  and  $\delta > 0$  so that:

1.  $\mathbb{E}[Z_{G,\eta'}(\lambda)] \leq e^{-\delta n} \mathbb{E}[Z_{G,\eta}(\lambda)]$  for all  $\eta'$  such that  $|\eta' - \eta| > \epsilon$ .
2.  $\mathbb{E}[Z_{G,\eta'}(\lambda)] \leq e^{-\delta n} \mathbb{E}[Z_{G,\eta_-}(\lambda)]$  for all  $\eta'$  such that  $|\eta' - \eta_-| \in (\epsilon, 2\epsilon)$ .

Next, we sketch how we construct the configuration sets  $S_1, S_2, S_3$  for applying Lemma 19, where we assume here for simplicity that the magnetization  $\eta$  can actually be realized on  $G$ . For  $\epsilon > 0$  as above, we set:

$S_1$ : configurations with magnetization  $\eta$

$S_2$ : configurations with magnetization in  $[\eta_- - \epsilon, \eta_- + \epsilon]$

$S_3$ : configurations with magnetization in  $[\eta_- - 2\epsilon, \eta_- - \epsilon] \cup (\eta_- + \epsilon, \eta_- + 2\epsilon]$ .

First, note that the Glauber dynamics starting in  $S_2$  must pass through  $S_3$  to reach  $S_1$ . Abbreviating  $\mu_{G,\beta,\lambda}$  as  $\mu$ , we can use Lemma 31, Theorem 32 and Property 1 from above to show that  $\mu(S_2) < \mu(S_1)$  a.a.s. over  $G$ . Similarly, using Proposition 30, Property 2 and Proposition 33 yields  $\mu(S_3) \leq e^{-\Omega(n)} \mu(S_2)$  a.a.s. Hence, applying Lemma 19, we conclude that the mixing time of Glauber dynamics on  $G$  is  $\exp(\Omega(n))$ .

### Slow Mixing of the Kawasaki Dynamics

We proceed with sketching the proof of part (2) of Theorem 1. Let  $\beta > \beta_u(\Delta)$ . We consider a graph  $H$  consisting of  $m$  identical copies  $G_1, G_2, \dots, G_m$  of a random  $\Delta$ -regular graph  $G$  from  $\mathbf{G}$ , where  $m$  is determined later based on  $\eta$ . We will separately consider the cases of  $|\eta| \in (\eta_c, \eta_u)$  and  $|\eta| \leq \eta_c$ , and assume without loss of generality that  $\eta > 0$ .

We start with the case  $\eta \in (\eta_c, \eta_u)$ . By Proposition 5, there exists  $\lambda_\eta \in (1, \lambda_u)$  such that  $\eta = \eta_{\Delta,\beta,\lambda_\eta}^+$ . For  $\lambda_+ \in (\lambda_\eta, \lambda_u)$ , set  $\eta_+ = \eta_{\Delta,\beta,\lambda_+}^+$  and  $\eta_- = \eta_{\Delta,\beta,\lambda_+}^-$ . In particular, note that we may choose  $\lambda_+$  such that there are  $m, \ell \in \mathbb{N}$  with  $\ell < m$  and  $m\eta = \ell\eta_+ + (m - \ell)\eta_-$ , where  $m$  is used for constructing  $H$ . Further, observe that  $\eta$  is the global maximizer of  $f_{\Delta,\beta,\lambda_\eta}$  and that  $\eta_+$  and  $\eta_-$  are the global and local maximizers of  $f_{\Delta,\beta,\lambda_+}$ . Hence, there are  $\epsilon > 0$  and  $\delta > 0$  so that:

1.  $\mathbb{E}[Z_{G,\eta'}(\lambda_\eta)] \leq e^{-\delta n} \mathbb{E}[Z_{G,\eta}(\lambda_\eta)]$  for all  $\eta'$  such that  $|\eta' - \eta| > \epsilon$ .
2.  $\mathbb{E}[Z_{G,\eta'}(\lambda_+)] \leq e^{-\delta n} \mathbb{E}[Z_{G,\eta_+}(\lambda_+)]$  for all  $\eta'$  such that  $|\eta' - \eta_+| \in (\epsilon, 2\epsilon)$ .
3.  $\mathbb{E}[Z_{G,\eta'}(\lambda_+)] \leq e^{-\delta n} \mathbb{E}[Z_{G,\eta_-}(\lambda_+)]$  for all  $\eta'$  such that  $|\eta' - \eta_-| \in (\epsilon, 2\epsilon)$ .

As for proving slow mixing of Glauber dynamics, we aim for applying Lemma 19. To sketch the construction of  $S_1, S_2, S_3$ , we again assume here for simplicity that a magnetization of  $\eta$  can be realized on each subgraph  $G_i$ . Given a configuration, we write  $\eta_{G_i}$  for the magnetization on subgraph  $G_i$ . We then take the following subsets of configurations on  $H$  with overall magnetization  $\eta$ :

$S_1$ :  $\eta_{G_i} = \eta$  for all  $1 \leq i \leq m$ ,

$S_2$ :  $\eta_{G_i} \in [\eta_+ - \epsilon, \eta_+ + \epsilon]$  for all  $i \leq \ell$  and  $\eta_{G_i} \in [\eta_- - \epsilon, \eta_- + \epsilon]$  for all  $i > \ell$ ,

$S_3$ :  $\eta_{G_i} \in [\eta_+ - 2\epsilon, \eta_+ + \epsilon]$  for all  $i \leq \ell$  and  $\eta_{G_i} \in [\eta_- - \epsilon, \eta_- + 2\epsilon]$  for all  $i > \ell$ , and there exists  $i \leq \ell$  with  $\eta_{G_i} \in [\eta_+ - 2\epsilon, \eta_+ - \epsilon]$  or  $i > \ell$  with  $\eta_{G_i} \in [\eta_- + \epsilon, \eta_- + 2\epsilon]$ .

Note that the Kawasaki dynamics have to pass through  $S_3$  to get from  $S_2$  to  $S_1$ . Moreover, abbreviating  $\hat{\mu}_{H,\beta,k}$  as  $\hat{\mu}$ , we can use Theorem 32, Lemma 31 and Property 1 to show that  $\hat{\mu}(S_1) \geq \hat{\mu}(S_2)$ , and we can use Lemma 31, Properties 2 and 3, Theorem 32 and Proposition 33 to show that  $\hat{\mu}(S_3) \leq e^{-\Theta(n)} \hat{\mu}(S_2)$  a.s. Hence, applying Lemma 19, we conclude that the mixing time of Kawasaki dynamics on  $H$  is  $\exp(\Omega(n))$ .

In the case that  $0 < \eta \leq \eta_c$ , we require a slightly different argument since we cannot apply Proposition 5 to  $\eta$ . Instead, we argue that for all  $\eta \in (0, \eta_c]$  we can choose  $\delta' > 0$  sufficiently small such that for all  $\eta_+ \in (\eta_c, \eta_c + \delta')$  and  $\eta_- = \eta_{\Delta, \beta, \lambda_{\eta_+}}^-$  it holds that  $\eta_- < \eta < \eta_+$ . In particular, we may choose  $\eta_+$  such that  $m\eta = \ell\eta_+ + (m - \ell)\eta_-$  for some  $m, \ell \in \mathbb{N}, \ell < m$ . We then define  $S_1, S_2, S_3$  (again with some slight simplification here) by

$$\begin{aligned} S_1: \eta_{G_i} &\in [\eta_- - \epsilon, \eta_- + \epsilon] \text{ for all } i \leq m - \ell \text{ and } \eta_{G_i} \in [\eta_+ - \epsilon, \eta_+ + \epsilon] \text{ else,} \\ S_2: \eta_{G_i} &\in [\eta_+ - \epsilon, \eta_+ + \epsilon] \text{ for all } i \leq \ell \text{ and } \eta_{G_i} \in [\eta_- - \epsilon, \eta_- + \epsilon] \text{ else,} \\ S_3: \eta_{G_i} &\in [\eta_+ - 2\epsilon, \eta_+ + \epsilon] \text{ for all } i \leq \ell \text{ and } \eta_{G_i} \in [\eta_- - \epsilon, \eta_- + 2\epsilon] \text{ else, and there} \\ &\text{exists } i \leq \ell \text{ with } \eta_{G_i} \in [\eta_+ - 2\epsilon, \eta_+ - \epsilon] \text{ or } i > \ell \text{ with } \eta_{G_i} \in [\eta_- + \epsilon, \eta_- + 2\epsilon]. \end{aligned}$$

By symmetry, we have  $\hat{\mu}(S_1) = \hat{\mu}(S_2)$  and by the same arguments as before it holds that  $\hat{\mu}(S_3) \leq e^{-\Theta(n)} \hat{\mu}(S_2)$ . Applying Lemma 19 then gives the desired result.

---

## References

- 1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1198–1211, 2020. doi:10.1145/3357713.3384317.
- 2 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *SIAM Journal on Computing*, 0(0):FOCS20-1, 2021. doi:10.1137/20M1367696.
- 3 Victor Bapst and Amin Coja-Oghlan. Harnessing the Bethe free energy. *Random structures & algorithms*, 49(4):694–741, 2016. doi:10.1002/rsa.20692.
- 4 Roland Bauerschmidt, Thierry Bodineau, and Benoit Dagallier. Kawasaki dynamics beyond the uniqueness threshold. *arXiv preprint arXiv:2310.04609*, 2023. doi:10.48550/arXiv.2310.04609.
- 5 Roland Bauerschmidt, Thierry Bodineau, and Benoit Dagallier. Stochastic dynamics and the Polchinski equation: an introduction. *arXiv preprint arXiv:2307.07619*, 2023. doi:10.48550/arXiv.2307.07619.
- 6 Rodney J Baxter. *Exactly solved models in statistical mechanics*. Elsevier, 2016.
- 7 Russ Bubley and Martin Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 223–231. IEEE, 1997. doi:10.1109/SFCS.1997.646111.
- 8 Van Hao Can, Remco van der Hofstad, and Takashi Kumagai. Glauber dynamics for Ising models on random regular graphs: cut-off and metastability. *ALEA*, 18(1):1441–1482, 2021. doi:10.30757/ALEA.v18-52.
- 9 N Cancrini and F Martinelli. On the spectral gap of Kawasaki dynamics under a mixing condition revisited. *Journal of Mathematical Physics*, 41(3):1391–1423, 2000. doi:10.1063/1.533192.
- 10 N Cancrini, F Martinelli, and C Roberto. The logarithmic Sobolev constant of Kawasaki dynamics under a mixing condition revisited. *Annales de l'Institut Henri Poincaré (B) Probability and Statistics*, 38(4):385–436, 2002. doi:10.1016/S0246-0203(01)01096-2.
- 11 Nicoletta Cancrini, F Cesi, and F Martinelli. The spectral gap for the Kawasaki dynamics at low temperature. *Journal of statistical physics*, 95:215–271, 1999. doi:10.1023/A:1004581512343.
- 12 Charlie Carlson, Ewan Davies, Alexandra Kolla, and Will Perkins. Computational thresholds for the fixed-magnetization Ising model. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1459–1472, 2022. doi:10.1145/3519935.3520003.
- 13 Raphaël Cerf and Ágoston Pisztora. On the Wulff crystal in the Ising model. *Annals of probability*, pages 947–1017, 2000. doi:10.1214/aop/1019160324.



- 14 Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for Markov chains. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–122. IEEE, 2022. doi:10.1109/FOCS54457.2022.00018.
- 15 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*, pages 1537–1550, 2021. doi:10.1145/3406325.3451035.
- 16 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Spectral independence via stability and applications to holant-type problems. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 149–160. IEEE, 2022. doi:10.1109/FOCS52979.2021.00023.
- 17 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. *SIAM Journal on Computing*, 52(1):196–237, 2023. doi:10.1137/20M136685X.
- 18 Amin Coja-Oghlan, Charilaos Efthymiou, and Samuel Hetterich. On the chromatic number of random regular graphs. *Journal of Combinatorial Theory, Series B*, 116:367–439, 2016. doi:10.1016/j.jctb.2015.09.006.
- 19 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. *Communications in Mathematical Physics*, pages 1–41, 2023. doi:10.1007/s00220-023-04644-6.
- 20 Amin Coja-Oghlan, Florent Krzakala, Will Perkins, and Lenka Zdeborová. Information-theoretic thresholds from the cavity method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 146–157, 2017. doi:10.1145/3055399.3055420.
- 21 Amin Coja-Oghlan, Philipp Loick, Balázs F Mezei, and Gregory B Sorkin. The Ising antiferromagnet and max cut on random regular graphs. *SIAM Journal on Discrete Mathematics*, 36(2):1306–1342, 2022. doi:10.1137/20M137999X.
- 22 Ewan Davies and Will Perkins. Approximately counting independent sets of a given size in bounded-degree graphs. *SIAM Journal on Computing*, 52(2):618–640, 2023. doi:10.1137/21M1466220.
- 23 Amir Dembo and Andrea Montanari. Ising models on locally tree-like graphs. *Ann. Appl. Probab.*, 20(1):565–592, 2010. doi:10.1214/09-AAP627.
- 24 Roland Lvovich Dobrushin, Roman Kotecký, and Senya Shlosman. *Wulff construction: a global shape from local interaction*, volume 104. American Mathematical Society Providence, 1992.
- 25 Martin Dyer, Alan Frieze, and Mark Jerrum. On counting independent sets in sparse graphs. *SIAM Journal on Computing*, 31(5):1527–1541, 2002. doi:10.1137/S0097539701383844.
- 26 Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, and Russell Martin. Markov chain comparison. *Probability Surveys*, 3(none):89–111, 2006. doi:10.1214/154957806000000041.
- 27 Andreas Galanis, Qi Ge, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Structures & Algorithms*, 45(1):78–110, 2014. doi:10.1002/rsa.20479.
- 28 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. doi:10.1017/S0963548315000401.
- 29 Andreas Galanis, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Ferromagnetic Potts model: Refined #BIS-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016. doi:10.1137/140997580.
- 30 Hans-Otto Georgii. *Gibbs measures and phase transitions*, volume 9. Walter de Gruyter, 2011.
- 31 Antoine Gerschenfeld and Andrea Montanari. Reconstruction for models on random graphs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS’07)*, pages 194–204. IEEE, 2007. doi:10.1109/FOCS.2007.58.

- 32 Heng Guo and Pinyan Lu. Uniqueness, spatial mixing, and approximation for ferromagnetic 2-spin systems. *ACM Transactions on Computation Theory (TOCT)*, 10(4):1–25, 2018. doi:10.1145/3265025.
- 33 Vishesh Jain, Marcus Michelen, Huy Tuan Pham, and Thuy-Duong Vuong. Optimal mixing of the down-up walk on independent sets of a given size. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1665–1681. IEEE, 2023. doi:10.1109/FOCS57990.2023.00101.
- 34 Svante Janson, Tomasz Łuczak, and Andrzej Rucinski. *Random graphs*. John Wiley & Sons, 2011.
- 35 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on computing*, 22(5):1087–1116, 1993. doi:10.1137/0222066.
- 36 Aiya Kuchukova, Marcus Pappik, Will Perkins, and Corrine Yap. Fast and slow mixing of the kawasaki dynamics on bounded-degree graphs. *arXiv preprint arXiv:2405.06209*, 2024. doi:10.48550/arXiv.2405.06209.
- 37 David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.
- 38 Sheng Lin Lu and Horng-Tzer Yau. Spectral gap and logarithmic Sobolev inequality for Kawasaki and Glauber dynamics. *Communications in Mathematical Physics*, 156(2):399–433, 1993. doi:10.1007/BF02098489.
- 39 Fabio Martinelli, Alistair Sinclair, and Dror Weitz. Fast mixing for independent sets, colorings, and other models on trees. *Random Structures & Algorithms*, 31(2):134–172, 2007. doi:10.1002/rsa.20132.
- 40 Elchanan Mossel and Allan Sly. Exact thresholds for Ising–Gibbs samplers on general graphs. *The Annals of Probability*, 41(1):294–328, 2013. doi:10.1214/11-AOP737.
- 41 Elchanan Mossel, Dror Weitz, and Nicholas Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probability Theory and Related Fields*, 143(3-4):401–439, 2009. doi:10.1007/s00440-007-0131-9.
- 42 Dana Randall and David Wilson. Sampling spin configurations of an Ising system. In *Symposium on Discrete Algorithms: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 959–960, 1999. doi:10.1145/314500.314945.
- 43 Guus Regts. Absence of zeros implies strong spatial mixing. *Probability Theory and Related Fields*, pages 1–21, 2023. doi:10.1007/s00440-023-01190-z.
- 44 Shuai Shao and Yuxin Sun. Contraction: A unified perspective of correlation decay and zero-freeness of 2-spin systems. *Journal of Statistical Physics*, 185:1–25, 2021. doi:10.1007/s10955-021-02831-0.
- 45 Shuai Shao and Xiaowei Ye. From zero-freeness to strong spatial mixing via a christoffel-darboux type identity. *arXiv preprint arXiv:2401.09317*, 2024. doi:10.48550/arXiv.2401.09317.
- 46 Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 287–296. IEEE, 2010. doi:10.1109/FOCS.2010.34.
- 47 Allan Sly and Nike Sun. Counting in two-spin models on d-regular graphs. *Annals of Probability*, 42(6):2383–2416, 2014. doi:10.1214/13-AOP888.
- 48 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. doi:10.1145/1132516.1132538.
- 49 Lawrence Zalcman. Normal families: new perspectives. *Bulletin of the American Mathematical Society*, 35(3):215–230, 1998. doi:10.1090/S0273-0979-98-00755-1.




# Stochastic Distance in Property Testing

Uri Meir   

Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Gregory Schwartzman  

JAIST, Japan

Yuichi Yoshida   

Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan

---

## Abstract

We introduce a novel concept termed “stochastic distance” for property testing. Diverging from the traditional definition of distance, where a distance  $t$  implies that there exist  $t$  edges that can be added to ensure a graph possesses a certain property (such as  $k$ -edge-connectivity), our new notion implies that there is a *high probability* that adding  $t$  *random* edges will endow the graph with the desired property. While formulating testers based on this new distance proves challenging in a sequential environment, it is much easier in a distributed setting. Taking  $k$ -edge-connectivity as a case study, we design ultra-fast testing algorithms in the CONGEST model. Our introduction of stochastic distance offers a more natural fit for the distributed setting, providing a promising avenue for future research in emerging models of computation.

**2012 ACM Subject Classification** Theory of computation → Streaming, sublinear and near linear time algorithms; Mathematics of computing → Random graphs; Theory of computation → Distributed algorithms

**Keywords and phrases** Connectivity,  $k$ -edge connectivity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.57

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.14080>

**Funding** *Gregory Schwartzman*: This work was supported by JSPS KAKENHI Grant Number JP21K17703 and JP21H05850.

*Yuichi Yoshida*: This work was supported by JSPS KAKENHI Grant Number 20H05965 and 22H05001.

## 1 Introduction

Property testing has become a major focus in computational research over the years. One of the main goals in this field is to quickly determine if a given structure has a specific property or is far from having it. Traditionally, this “distance” from a property has been defined using the Hamming distance, which counts the number of changes needed to give the structure the desired property. But a question arises: Is this the best way to measure distance in all situations, especially in emerging models of computation?

Consider the following motivating example. Distributed dynamic systems, such as peer-to-peer networks, frequently experience changes in their structure as nodes (clients) join or depart and edges may experience failures. For the sake of resilience against failures, it is imperative for these systems to maintain a topology with certain advantageous features, like  $k$ -edge-connectivity<sup>1</sup>. However, maintaining this throughout the evolution of the network

---

<sup>1</sup> Going forward we simply write  $k$ -connectivity.



may require resource-intensive corrections, which ideally should be minimized. Still, it might be the case that some topologies are “easy” to correct, in the sense that simply adding a small number of random edges to the graph will make the network topology  $k$ -connected. This can be seen as a “low-cost” fixing operation, compared to executing an elaborate algorithm that guarantees  $k$ -connectivity for *every* topology. A question arises: can we detect these easy-to-fix topologies *fast*?

We take a first step in addressing the above and introduce a new distance measure which we call *stochastic distance*. That is, we say that a graph  $G = (V, E)$  is  $t$ -stochastically-close to a property  $\mathcal{P}$  if it holds with high probability (w.h.p.)<sup>2</sup> that adding (roughly)  $t$  random edges to  $G$  will make it have property  $\mathcal{P}$ . Intuitively, the Hamming distance metric can be seen as asking whether the input graph is close to *at least one* graph instance that has a desired property, while our new distance measure asks whether the graph is close to *many instances* that have the property.

To exemplify the usefulness of our new distance measure, we take the property of  $k$ -connectivity as a case study. This is an extremely desirable graph property in the distributed setting, as it guarantees that the system remains connected even under several edge failures. In Section 3 we note that the simple case of connectivity already becomes hard to test in the sequential setting. However, using the distributed power of the system we can design extremely fast distributed testers for both connectivity and  $k$ -connectivity.

## 1.1 Our model and results

In the distributed setting, a network of nodes, which is represented by a communication graph  $G = (V, E)$ , aims to solve some graph problem with respect to  $G$ . Every node in  $G$  has unique ID of  $O(\log n)$  bits. Computation proceeds in synchronous rounds, in each of which every vertex can send a message to each of its neighbors. The running time of the algorithm is measured as the number of communication rounds it takes to finish. Our results hold for the CONGEST model of distributed computation, where messages are limited to  $O(\log n)$  bits (where  $n = |V|$ ).

A distributed 1-sided tester [5] for a property  $\mathcal{P}$  (or simply a tester) has the following guarantee. If the graph  $G$  has the property, then all nodes accept. If the graph is  $\epsilon$ -far from having the property, then with probability larger than  $2/3$  *at least* one node rejects. This definition remains the same for both the Hamming distance metric and our stochastic distance measure.

Note that  $\epsilon \in (0, 1)$  indicates the distance from the property relative to the number of edges in the graph (i.e.,  $\epsilon|E|$  in the general model) or to the possible number of edges in the graph (i.e.,  $\epsilon \binom{n}{2}$  in the dense model). In our model every non-edge is added with probability  $t/(\binom{n}{2} - |E|)$  (see Section 2 for an exact definition), therefore it is more natural to use  $t \in \mathbb{N}$  as the distance parameter rather than  $\epsilon$ . That is, our testers decide whether the graph has a certain property or if it is  $t$ -far from it. We state and prove the following two theorems<sup>3</sup>:

► **Theorem 1.** *There exists a deterministic algorithm in the CONGEST model, that for a parameter  $s \in \mathbb{N}$  runs in  $O(s)$  rounds and distinguishes whether the graph  $G$  is connected, or is  $\Omega((n \log n)/s)$ -stochastically-far.*

<sup>2</sup> With probability at least  $1 - n^{-c}$  for some constant  $c > 1$ . The choice of  $c$  does not affect the asymptotics of our results.

<sup>3</sup> Where  $\tilde{O}$  subsumes factors logarithmic in  $n$ .



► **Theorem 2.** *There exists a randomized algorithm in the CONGEST model, that for a parameter  $s \in \mathbb{N}$  runs in  $\tilde{O}(s^4)$  rounds and distinguishes w.h.p whether the graph  $G$  is  $k$ -connected, or is  $\Omega((kn \log n)/s)$ -stochastically-far from being one.*

Intuitively, the above states that the more random edges are added the easier it is to check if the graph will become  $k$ -connected or not. That is, if  $s \approx kn \log n$  then checking whether a constant number of random edges will make the graph  $k$ -connected is as hard as checking if the graph is connected, which requires traversing the entire graph. If  $s = O(1)$ , then so many edges are added that the graph almost surely becomes  $k$ -connected. Our results provide a smooth transition between these two cases.

## Related work

In (non-distributed) property testing, the objective is to devise algorithms that can distinguish between graphs satisfying a property  $\mathcal{P}$  and graphs that are  $\epsilon$ -far from having the property with a high probability. Testing connectivity properties in the bounded-degree model, where we have query access to the input graph via a list of incidence lists, has been extensively studied, including  $k$ -connectivity [13, 21, 17],  $k$ -vertex-connectivity [22, 17],  $(k, l)$ -sparsity [14] and supermodular-cut conditions [19].

The first distributed property testing algorithm was due to [3]. A thorough study of distributed property testing was initiated in [5]. This was followed by a long line of work, presenting new and improved testers for various properties [1, 9, 12, 11, 16, 10]. All of these works consider the Hamming distance handed down from the sequential testing model.

$k$ -connectivity received a large amount of attention in the distributed literature [20, 4, 8, 7, 6, 18, 2]. There is a large body of work that aims to find *sparse connectivity certificates* – for a  $k$ -connected graph the goal is to find a  $k$ -connected subgraph that has  $O(kn)$  edges. Another related problem is computing a  $k$ -edge-connected spanning subgraph ( $k$ -ECSS) – for a  $k$ -connected graph the goal is to find the *sparsest possible*  $k$ -connected subgraph. We note that both these problems assume that the input graph is  $k$ -connected, and are therefore inherently different than the problem we consider.

## 2 Preliminaries

### Distance from a property

We identify simple graphs  $G = (V = [n], E)$  with the  $\binom{n}{2}$ -dimensional characteristic vector of their edge set  $E \subseteq \binom{V}{2}$ , and use the Hamming metric over these vectors, defined by  $d_{\text{HAM}}(G, G') = \left| \{i \in \binom{V}{2} : G(i) \neq G'(i)\} \right|$ , where  $G(i)$  is the  $i^{\text{th}}$  entry in the vector  $G$ .

By extension, the distance of a graph  $G$  from a family of graphs  $\mathcal{G}$  is:

$$d_{\text{HAM}}(G, \mathcal{G}) = \min_{G' \in \mathcal{G}} d_{\text{HAM}}(G, G').$$

A property  $\mathcal{P}$  is formally a family of graphs (e.g., all connected graphs). Given positive integers  $n$  and  $t$ , we define the YES case of the corresponding testing problem to be all graphs  $G = (V, E)$  in  $\mathcal{P}$  over  $n$  vertices:

$$\text{YES} := \{G \in \{0, 1\}^{\binom{V}{2}} : G \in \mathcal{P}\}$$

We define the NO case, denoted  $\text{NO}'_t$ , to be all graphs  $G = (V, E)$  over  $n$  vertices with Hamming distance at least  $t$  from YES:

$$\text{NO}'_t := \{G \in \{0, 1\}^{\binom{V}{2}} : d_{\text{HAM}}(G, \text{YES}) \geq t\}$$

Motivated by connectivity, in the following we only consider monotone non-decreasing properties, for which it is easy to see that only additions count towards the distance from the property (note that many properties in the literature are monotone non-increasing instead. For these, one can replace additions with deletions). Formally, if  $\mathcal{P}$  is a monotone non-decreasing property, and  $G \notin \mathcal{P}$  is a graph that does not satisfy it, then for any  $H \in \mathcal{P}$  with  $d_{\text{HAM}}(G, H) = d_{\text{HAM}}(G, \mathcal{P})$ , we have  $E_G \subseteq E_H$  (or alternatively,  $d_{\text{HAM}}(G, H) = \left| \{i \in \binom{V}{2} : G(i) = 0 \wedge H(i) = 1\} \right|$ ).

### Stochastic distance

We define *stochastic closeness* to a property as follows:

► **Definition 3** (Random addition of edges). *For a graph  $G = (V, E)$ , we choose a random subset  $E'$  of the edge set  $\bar{E} = \binom{V}{2} \setminus E$ , by adding each edge with probability  $t/|\bar{E}|$  independently, for parameter  $t \in [0, |\bar{E}|]$ . We define the random graph  $\text{Add}(G, t) := (V, E \cup E')$ . We say that  $G' = \text{Add}(G, t)$  is created from  $G$  by a random addition of edges with parameter  $t$ .*

The parameter  $t$  should be understood intuitively as the (expected) number of random edges required to make the graph have the property  $\mathcal{P}$  w.h.p. Per our motivation, and in accordance with previous results for testing connectivity, we allow ourselves the mild assumption that  $|\bar{E}| = \Omega(n^2)$  which clearly holds for any input relevant to our setting. The amount of edges we aim to add, however, is always significantly smaller,  $t = o(n^2)$ . In particular it is always the case that  $t \leq |\bar{E}|$ , as desired.

► **Definition 4** (Stochastic closeness to a monotone property). *For a monotone property  $\mathcal{P}$ , a graph  $G \notin \mathcal{P}$  over  $n$  vertices is said to be  $t$ -stochastically-close to satisfying  $\mathcal{P}$  if the graph  $G' = \text{Add}(G, t)$  satisfies*

$$\Pr[G' \notin \mathcal{P}] \leq n^{-c}$$

for some global constant  $c > 1$ . As shown in Appendix A, the constant  $c$  can be chosen arbitrarily without affecting stochastic closeness by more than a constant factor.

We are now able to define the alternative set of NO instances:

$$\text{NO}_t := \{G \in \{0, 1\}^{\binom{V}{2}} : G \notin \text{YES and } G \text{ is not } t\text{-stochastically-close to } \mathcal{P}\}$$

In this text our main focus is solving promise problems of type  $(\text{YES}, \text{NO}_t)$ , rather than  $(\text{YES}, \text{NO}'_t)$ .

### Comparing the two notions of distance

As an illustrative example, consider two  $n$ -node graphs, both with exactly two connected components: in  $G_1$  a constant-size component is disconnected from the rest of the graph, while in  $G_2$  there are two components of the same size,  $n/2$ . While both graphs are exactly one edge away from being connected (i.e., both have hamming distance 1). The situation is quite different if edges are added randomly, instead of being handpicked, leading to different *stochastic* distance. While a small number of edges ( $\sim \log n$ ) are already likely to connect  $G_2$ , the same amount has only probability  $o(1)$  to connect  $G_1$ .

### 3 Warm-up: Connectivity with Stochastic Distance

In this section we deal with the graph connectivity property for a network that is not necessarily connected. This section aims to present the ideas used in the following section about  $k$ -connectivity. We use the following terminology: when a graph is disconnected, write  $G = \bigcup_{i=1}^m C_i$  as the unique decomposition of  $G$  into its connected components  $C_1, \dots, C_m$ , with  $C_i = (V_i, E_i)$ , and  $s_i := |V_i|$  for their sizes.

We make the following observation:

► **Observation 5.** *Let  $G = \bigcup_{i=1}^m C_i$  be a disconnected graph (i.e.,  $m \geq 2$ ), and let  $s = \frac{1}{m} \sum_{i \in [m]} s_i$ , then  $G$  is  $O(\frac{n}{s})$ -close to being connected in Hamming distance.*

Indeed, it holds that  $s = \frac{1}{m} \sum_{i \in [m]} s_i = n/m$ , which means that  $m = n/s$ . As a single edge can be used to connect two components, adding  $m - 1 = n/s - 1$  edges suffices to connect the graph.

The observation is also tight: there exist graphs that require exactly this number of additions. Joined with a Markovian argument, one can deduce that for any graph that is far from connectivity, there exists *many* small connected components. This is a key argument in the analysis of existing connectivity testers (using Hamming distance) [13].

The main part of this section deals with proving a statement of similar taste for stochastic distance. While the Hamming distance of a graph from being connected is dictated by the *average* size of a connected component, for stochastic distance this is dictated by the *minimum* size of a connected component. As a result, a graph that is far from being connected in stochastic distance is only guaranteed to have *one* small connected component. Formally, we prove the following lemma:

► **Lemma 6.** *Let  $G = \bigcup_{i=1}^m C_i$  be a disconnected graph (that is,  $m \geq 2$ ), with components of sizes  $s_i = |V_i|$ . Then  $G$  is  $O((n \log n)/s)$ -stochastically-close to being connected, where  $s = \min_{i \in [m]} \{s_i\}$ .*

**Proof.** Recall that when considering stochastic distance every non-edge is added with some probability  $p$ . Consider the set of bad events  $\{B_k\}_{k=1}^{\lfloor m/2 \rfloor}$ , where  $B_k$  is the event that there exists a set of exactly  $k$  connected components  $C_{i_1}, \dots, C_{i_k}$  such that none of the edges between  $\bigcup_{j=1}^k C_{i_j}$  and the rest of the graph are added. The important observation is that the graph stays disconnected if and only if one of these bad events occurs. We go on to bound the probability of each of these bad events.

Fix  $k$ . Our goal is to bound  $\Pr[B_k]$ . There are  $\binom{m}{k}$  ways to choose a subset of  $k$  components. Fix one such choice  $i_1, \dots, i_k$  and denote the set of vertices in these components by  $U = \bigcup_{j=1}^k V_{i_j}$ . As each component is of size at least  $s$ , we have  $|U| \geq ks$ , applying the same reasoning to the remaining graph, made of the rest of the components:  $|V \setminus U| \geq (m - k)s \geq ks$  nodes (the second inequality holds since  $k \leq m/2$ ). Thus, we have  $ks \leq |U| \leq n - ks$ , and consequently we get:

$$|U| \cdot |V \setminus U| = |U| \cdot (n - |U|) \geq ks(n - ks).$$

where the inequality is true since  $f(x) = x(n - x)$  is unimodal and symmetric on the interval  $[ks, n - ks]$ . The probability of  $U$  staying disconnected from  $V \setminus U$  is thus bounded by:

$$(1 - p)^{|U|(n - |U|)} \leq e^{-p|U|(n - |U|)} \leq e^{-pks(n - ks)}$$

And by a union bound over all choices of  $U$  with  $k$  connected components, we get:

$$\Pr[B_k] \leq \binom{m}{k} e^{-pks(n - ks)} \leq e^{-pks(n - ks) + k \log m} \leq e^{-pksn + p(ks)^2 + k \log n}$$

Thus, by taking  $p = 2(c+2) \log n / (sn)$ , the expression in the exponent is bounded by:

$$\begin{aligned} -\frac{2(c+2) \log n \cdot ksn}{sn} + \frac{2(c+2) \log n (ks)^2}{sn} + k \log n &= (2(c+2)(ks/n - 1) + 1)k \log n \\ &\leq -(c+1)k \log n \end{aligned}$$

where the inequality uses  $ks/n \leq (m/2)s/n \leq 1/2$ . This, in turn, bounds the probability of the bad event by:

$$\Pr[B_k] \leq e^{-(c+1)k \log n} \leq \left(\frac{1}{n}\right)^{-(c+1)k}$$

Using a union bound over all values of  $k$ , we get

$$\Pr\left[\bigvee_{k=1}^{\lfloor m/2 \rfloor} B_k\right] \leq \sum_{k=1}^{\lfloor m/2 \rfloor} n^{-(c+1)k} \leq \sum_{k=1}^{\infty} n^{-(c+1)k} \leq 2n^{-(c+1)} \leq n^{-c}$$

where the second to last inequality uses the sum of an infinite geometric series with common ratio  $n^{-(c+1)} \leq 1/2$ , and the last simply uses  $n \geq 2$ .

Finally, we get that  $G$  is connected w.h.p. This implies that  $G$  is  $t$ -stochastically-close to being connected, with  $t = O((n \log n)/s)$  (recall that  $|\bar{E}| = \Omega(n^2)$ ). ◀

### Tightness of Lemma 6

We focus on the smallest connected component,  $V_i$  of size  $s_i = s$ . There are at most  $s(n-s) \leq sn$  potential edges to connect  $V_i$  to the rest of the graph. If we take  $p' = c \log n / (4sn)$ , then the probability none of them is added satisfies:

$$(1 - p')^{s(n-s)} \geq (1 - p')^{sn} \geq e^{-2p'sn} = e^{-c \log n / 2} = n^{-c/2} > n^{-c},$$

where the second inequality uses  $1 - x \geq e^{-2x}$ , which holds for  $x \in [0, 1/2]$ .

Using the counter-positive of Lemma 6, a graph  $G$  that is  $O((n \log n)/s)$ -stochastically-far from being connected is guaranteed to have a connected component of size *at most*  $O(s)$ .

It is clear that a sequential tester is ill-suited to detect such small witnesses. Indeed, consider distinguishing the connected cycle over all  $n$  nodes, from a disconnected  $n$ -node graph consisting of an  $n - 1$  cycle and a single isolated node (chosen uniformly at random). On the one hand, all disconnected graphs are as stochastically far as a graph can be from connectivity (requiring roughly  $n \log n$  random edge additions). Still, a sequential tester would require  $\Omega(n)$  queries to an oracle in order to detect the isolated node.

In the CONGEST model, however, an efficient testing procedure exists.

► **Theorem 1.** *There exists a deterministic algorithm in the CONGEST model, that for a parameter  $s \in \mathbb{N}$  runs in  $O(s)$  rounds and distinguishes whether the graph  $G$  is connected, or is  $\Omega((n \log n)/s)$ -stochastically-far.*

**Proof.** Assume that the parameter  $s$  is known to all nodes in the network. Each node runs a distributed DFS algorithm. Every execution is associated with an ID, which is simply the ID of the root of the DFS tree. When multiple DFS executions visit the same node, all execution are terminated, except that with the maximum ID. When a DFS execution visits  $s$  nodes, it terminates and checks whether there is an outgoing edge from any visited node to any unvisited node. If there is, then it accepts (i.e., the graph is connected), otherwise it rejects (i.e., the graph is far from being connected). Correctness follows from Lemma 6. It is clear that there is no congestion and that the algorithm terminates in  $O(s)$  iterations. ◀

## 4 Stochastic Distance from $k$ -Connectivity

This section deals with  $k$ -connectivity, for any  $k \geq 1$ . We aim to characterize stochastic distance of a graph in terms of cuts, focusing on a specific parameter-of-interest denoted by  $s_k(G)$ : the smallest size of a vertex set that has a cut strictly smaller than  $k$ . In terms of this parameter, the following can be shown:

► **Theorem 7.** *Let  $G = (V, E)$  and  $s = s_k(G)$ , then  $G$  is  $O((k \cdot n \log n)/s)$ -stochastically-close to being  $k$ -connected.*

Similarly to Lemma 6, this theorem is tight, up to a factor of  $k$ . The rest of this section deals with proving the above theorem. First the parameter  $s_k(G)$  is formally defined and discussed. Later, we generalize Lemma 6 to bound the number of random edge additions required to increase the connectivity of a graph by one. In the last part we formally prove Theorem 7.

### 4.1 Properties of minimal cuts

For any vertex set,  $U \subseteq V$ , we write the cut size of  $U$  as  $c(U) := |E(U, V \setminus U)|$ . We use  $d(t) := t(n - t)$  for the *potential* amount of edges between any  $t$  vertices and the rest of the graph, and for a specific vertex set,  $U$ , we slightly abuse notation and write  $d(U) = d(|U|)$ .

For any parameter  $k \in \mathbb{N}$  and graph  $G = (V, E)$ , we define the collection of small cuts as the cuts that are strictly smaller than  $k$ :

$$\mathcal{S}_k(G) := \{U \subseteq V : c(U) < k \text{ and } U \neq \emptyset\}$$

A value of interest for us will be the size of the smallest set in such a collection. Formally we define:

$$s_k(G) := \min_{U \in \mathcal{S}_k(G)} |U|$$

When the connectivity parameter  $k$  and the graph  $G$  are clear from context, we omit either one or both (writing  $s_k, s(G)$  or simply  $s$  instead). If the collection is empty for some  $k$  and  $G$ , we define  $s_k(G) = n$ . Note that  $s_k(G) = n$  if and only if  $G$  is indeed  $k$ -connected.

We show that  $s_k(G)$  has certain monotonicity properties with respect to the parameters  $G$  and  $k$ . Indeed, by definition a larger value for  $k$  creates a larger collection of cuts, i.e.,  $\mathcal{S}_k(G) \subseteq \mathcal{S}_{k+1}(G)$ . On the other hand, adding edges to the graph can only increase the cut of any given vertex set,  $U$ . Thus, any  $U$  with a small cut after additions, also had a small cut before additions. Formally, if  $G \subseteq G'$  (i.e.,  $G = (V, E), G' = (V, E')$  and  $E \subseteq E'$ ), we have  $\mathcal{S}_k(G') \subseteq \mathcal{S}_k(G)$ . The minimum over elements in a collection can only decrease if we add elements to the collection, and hence we have:

► **Observation 8** (Monotonicity of  $s_k(G)$ ). *The parameter  $s_k(G)$  is monotone non-increasing in  $k$ , and monotone non-decreasing in  $G$ .*

Below, we will be interested in finding the smallest vertex set  $U$  in the collection  $\mathcal{S}_k(G)$ . This element determines the value  $s$ . The following proposition shows a property of such a set  $U$  that will be useful for the algorithm:

► **Proposition 9.** *Fix  $k \in \mathbb{N}$  and a graph  $G = (V, E)$ . For any subset  $U \subseteq V$  with a cut less than  $k$  ( $c(U) < k$ ) and of minimal size ( $|U| = s_k$ ) the induced subgraph  $G[U]$  is connected.*

**Proof.** Assume otherwise, then we can break  $G[U]$  into its connected components:  $U = \bigcup_i U_i$ . Each such component has strictly less nodes than  $U$ . Moreover, we have:

$$c(U_i) = E(U_i, V \setminus U_i) = E(U_i, V \setminus U) \leq E(U, V \setminus U) = c(U),$$

where the second equality holds since there are no edges between different components of  $U$ . We conclude that each  $U_i$  has a cut of size at most  $c(U) < k$ , but their size is strictly smaller than  $s$ , which leads to a contradiction. ◀

### 4.2 Increasing the connectivity by one

Using the notations of this section, the statement of Lemma 6 is that any graph that is “0-connected” (i.e., disconnected) is  $O(n \log(n)/s_1)$ -stochastically-close to being 1-connected. This is true since  $s_1$  is exactly the size of the smallest connected component in  $G$ .

We generalize this statement for  $k$ -connectivity:

► **Lemma 10.** *Any  $(k - 1)$ -connected graph  $G$  with  $n \geq 4k$  nodes is  $O(n \log(n)/s_k)$ -stochastically-close to being  $k$ -connected.*

**Proof.** For  $k = 1$ , the proof is complete due to Lemma 6. For  $k > 1$ , let  $G$  be a  $(k - 1)$ -connected graph. Thus, all minimum cuts have  $k - 1 \geq 1$  edges ( $G$  is connected). We estimate the number of random edges needed to make all sets in  $\mathcal{S}_k(G)$  have a cut of size at least  $k$ .

We identify minimum cut with a partition of  $V$  to  $U$  and  $V \setminus U$  (w.l.o.g,  $|U| \leq \lfloor n/2 \rfloor$ ). As in Lemma 6 assume that every non-edge is added with probability  $p$ . Consider the event  $B_U$  that  $U$  remains with cut of size  $k - 1$ . That is, not a single edge was added between  $U$  and  $V \setminus U$ . There are exactly  $d(U) - (k - 1)$  potential additions, and so:

$$\Pr[B_U] = (1 - p)^{d(U) - (k - 1)} = (1 - p)^{|U|(n - |U|) - (k - 1)} \leq (1 - p)^{s_k \cdot n/4} \leq e^{-s_k \cdot pn/4},$$

where the first inequality uses  $n - |U| \geq n/2$ ,  $|U| \geq s_k(G)$  and  $k \leq |U|n/4$  to lower bound the exponent.

By taking  $p = 4(c + 2) \log n / (s_k n)$ , this bound becomes

$$e^{-s_k \cdot pn/4} = e^{-(c+2) \log n} = n^{-(c+2)}$$

We finish the proof by using a union bound over all minimum cuts. There are at most  $\binom{n}{2} \leq n^2$  of them due to well known corollary of Karger’s Algorithm [15]. The probability that  $G$  is not  $k$ -connected after the edge additions is at most

$$\sum_{U \in \mathcal{S}_k(G)} \Pr[B_U] \leq n^2 \cdot n^{-(c+2)} = n^{-c}. \quad \blacktriangleleft$$

### 4.3 Putting it all together

We are now ready to prove Theorem 7. Our proof considers an alternative addition process which is easier to analyze. Intuitively, let  $p' \approx p/k$  and consider the following process. “Repeat  $k$  times: add each edge to the graph independently with probability  $p'$ ”. We formalize the above and bound the required value of  $p'$  for the alternative process, analyzing each of the  $k$  iterations separately with Lemma 10. We finish up by relating the original addition processes to the alternative one.

**Proof of Theorem 7.** Denote by  $r < k$  the connectivity of  $G$  (if  $G$  is disconnected, then  $r = 0$ ). We consider  $k - r$  (at most  $k$ ) iterations of additions, which we enumerate by  $r + 1, \dots, k$ . We further denote by  $G_i$  the graph after iteration  $i$  (and  $G_r = G$ ).

First we consider process (A), where at each iteration  $i \in \{r + 1, \dots, k\}$ , each edge is independently added with probability

$$p_i = 4(c + 3)n \log n / s_{i+1}(G_i).$$

Applying Lemma 10 for each iteration  $i$  we have the following: The probability that  $G_i$  is  $i$ -connected, *given that  $G_{i-1}$  is  $(i - 1)$ -connected* is at least  $1 - n^{-(c+1)}$ . Using a union bound over all  $k - r \leq n$  iterations, we have that with probability at least  $1 - n^{-c}$ , the final graph  $G_k$  is  $k$ -connected.

Next, consider an adjusted process (B) that also works in iterations, but uses a fixed value

$$p' = 4(c + 3)n \log n / s_k(G).$$

Using the monotonicity of  $s_k(G)$ , we have

$$s_k(G) = s_k(G_r) \leq s_{i+1}(G_r) \leq s_{i+1}(G_i),$$

and thus  $p_i \leq p'$  for all values of  $i$ .

Let us focus on a single non-edge of  $G$ , call it  $(u, v)$ , and follow it through the entire process. The probability  $(u, v)$  is added to  $G_k$  is higher in process (B) than it is in process (A), since  $p' \geq p_i$  for all  $i$ . On the other hand, this probability is at most  $p'/k$ , by a union bound over all iterations of process (B).

Lastly, consider process (C), a one-shot random addition, using

$$p = p'k = 4(c + 3)kn \log n / s_k(G).$$

The probability of every non-edge being added in process (C) is higher than it is in process (B) which is in turn higher than process (A). By monotonicity of  $k$ -connectivity (as a graph property), the probability that  $G_k$  is a  $k$ -connected graph is therefore also higher in process (C) than it is in process (A), and is therefore at least  $1 - n^{-c}$ . ◀

The counter-positive of Theorem 7 means that any graph  $G$  which is  $O((k \cdot n \log n)/s)$ -stochastically-far from being  $k$ -connected has a set  $W$  of at most  $s$  nodes with a cut smaller than  $k$ . We call such set  $W$  an  $(s, k)$ -witness, or simply a *witness* when  $k$  and  $s$  are clear from context.

## 5 Distributed algorithm for $k$ -connectivity

In this section, we provide a distributed algorithm that detects an  $(s, k)$ -witness within  $O(s^4 \log n)$  rounds in the CONGEST model w.h.p. We state with the following useful lemma<sup>4</sup>:

► **Lemma 11** (Lemma 3.16 in [13]). *Let  $W$  be an  $(s, k)$ -witness. Suppose that each edge in the graph is independently assigned a uniformly distributed cost in  $[0, 1]$ . Then, with probability at least  $\Theta(s^{-2(1-1/k)})$ ,  $W$  contains a spanning tree such that every edge in the tree has cost smaller than any edge in the cut  $E(W, V \setminus W)$ .*

<sup>4</sup> The original lemma in [13] uses the notion of a  $j$ -extreme node set which is equivalent to our notion of a  $(s, k)$ -witness.

## 57:10 Stochastic Distance in Property Testing

Specifically, if the tree guaranteed by the above lemma exists, the MST must also have the same guarantees. Furthermore, as all edge costs are unique w.h.p., the MST is unique w.h.p. When referring to the above lemma, it will be convenient to refer to a single tree, the MST induced by the cost function (i.e., when referring to the spanning tree guaranteed by Lemma 11, we always refer to an MST).

Given a starting vertex  $w$  and a size bound  $s$ , consider the following random process:

- Input: start vertex  $u \in V$ , size bound  $s \in \mathbb{N}$ .
- Assign uniformly random weights to all edges:  $w : E \rightarrow [0, 1]$ .
- Start with a singleton set  $W = \{u\}$ . As long as  $|W| < s$ , repeat the following:
  - Choose the smallest cut edge  $e = \operatorname{argmin}_{e \in E(W, V \setminus W)} w(e)$ .
  - Update  $W$  to include the new node of  $e$  in  $V \setminus W$ .
  - If  $c(W) < k$  then declare  $W$  as an  $(s, k)$ -witness.

Lemma 11 implies that repeating the above procedure  $\tilde{\Theta}(s^2)$  times detects a witness w.h.p.

In [13] a small amount of random nodes is sampled, and the above procedure is applied for each of them. This is possible since any graph that is far from  $k$ -connectivity in *Hamming* distance contains many witnesses. In our case, a graph that is *stochastically* far only guarantees a single witness, and therefore it is costly to detect the witness in the standard (query) model. We leverage the distributed nature of the system to apply the witness detection procedure to all nodes in parallel.

### Distributed implementation

The above algorithm admits a simple distributed implementation. First, we assign random weights in  $[0, 1]$  to all edges in the graph, this guarantees that with some small probability there exists a spanning tree within the witness component satisfying the guarantees of Lemma 11. Assume this tree exists and denote this tree by  $T$ . It will be clear from our algorithm that if this tree does not exist, the running time does not change however we may simply not detect the witness component if it exists.

Every node grows a cluster as follows. Every cluster can be implemented via a tree rooted at a leader node. All nodes in the cluster know for each neighbor whether it is in the cluster or not. To pick the next edge to be added to the cluster the edge weights are propagated along the tree towards the root, taking the minimum at every node. Finally, the root propagates the minimum value to all nodes in the cluster and the corresponding edge is added to the cluster. After each edge addition the cluster checks whether the current cut size is smaller than  $k$  and terminates if this is the case (a witness is found). For ease of analysis let us define an *iteration* as the step of adding a single edge (and vertex) to the cluster. When running this procedure from multiple nodes we can have every cluster wait for  $\Theta(i)$  rounds to complete the  $i$ -th iteration. This will guarantee when running this procedure from multiple nodes that the iterations are executed in lock-step. We execute this process for  $s$  iterations. This results in a running time of  $O(s^2)$  rounds.

In order to detect a witness component, we must execute the above algorithm to completion, starting from a node within the witness component. We do not know which nodes are within the witness component before executing the algorithm, and running the procedure to completion may cause congestion. To overcome this, we execute the procedure from all nodes simultaneously, but prioritize executions that overlap according to a priority condition.

The priority of the cluster at iteration  $i$  is the largest edge cost that was added to the cluster so far. Let us call this the *max-edge* value. Whenever multiple executions reach the same node, executions with higher max-edge values are terminated (ties are broken via



the cluster root IDs). As clusters add edges greedily according to edge weights, we know that only edges from  $T$  will be added to clusters whose execution starts from the witness component. That is, all edges in  $T$  will be added, at which point the small cut will be detected and all nodes in the cluster will reject.

As we terminate executions that visit the same node, we never experience congestion. It is sufficient to prove that at least one cluster execution that starts within the witness component survives. As all edges in the tree have weights smaller than the cut edges this means that the execution of clusters that start within the witness component is not affected by clusters that start from outside the component. This is because they always have a lower max-edge value due to guarantees on the weights of  $T$ . Furthermore, in every iteration at least one cluster in the witness component is not terminated. That is, the cluster with highest priority inside the witness component at iteration  $i$  must survive. We conclude that there exists a cluster construction that start within the component and terminates within  $O(s^2)$  rounds. To get the guarantee of Lemma 11 w.h.p., we must repeat the process  $\tilde{O}(s^2)$  times. Thus, the final running time is  $\tilde{O}(s^4)$ . We state the following theorem:

► **Theorem 2.** *There exists a randomized algorithm in the CONGEST model, that for a parameter  $s \in \mathbb{N}$  runs in  $\tilde{O}(s^4)$  rounds and distinguishes w.h.p whether the graph  $G$  is  $k$ -connected, or is  $\Omega((kn \log n)/s)$ -stochastically-far from being one.*

---

## References

- 1 John Augustine, Anisur Rahaman Molla, Gopal Pandurangan, and Yadu Vasudev. Byzantine connectivity testing in the congested clique. In *DISC*, volume 246 of *LIPICs*, pages 7:1–7:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 2 Marcel Bezdrighin, Michael Elkin, Mohsen Ghaffari, Christoph Grunau, Bernhard Haeupler, Saeed Ilchi, and Václav Rozhon. Deterministic distributed sparse and ultra-sparse spanners and connectivity certificates. In *SPAA*, pages 1–10. ACM, 2022.
- 3 Zvika Brakerski and Boaz Patt-Shamir. Distributed discovery of large near-cliques. *Distributed Comput.*, 24(2):79–89, 2011.
- 4 Keren Censor-Hillel and Michal Dory. Fast distributed approximation for TAP and 2-edge-connectivity. *Distributed Comput.*, 33(2):145–168, 2020.
- 5 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Comput.*, 32(1):41–57, 2019.
- 6 Mohit Daga, Monika Henzinger, Danupon Nanongkai, and Thatchaphol Saranurak. Distributed edge connectivity in sublinear time. In *STOC*, pages 343–354. ACM, 2019.
- 7 Michal Dory. Distributed approximation of minimum  $k$ -edge-connected spanning subgraphs. In *PODC*, pages 149–158. ACM, 2018.
- 8 Michal Dory and Mohsen Ghaffari. A nearly time-optimal distributed approximation of minimum cost  $k$ -edge-connected spanning subgraph. In *SODA*, pages 4296–4334. SIAM, 2023.
- 9 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In *DISC*, volume 91 of *LIPICs*, pages 15:1–15:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 10 Hendrik Fichtenberger and Yadu Vasudev. A two-sided error distributed property tester for conductance. In *MFCS*, volume 117 of *LIPICs*, pages 19:1–19:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 11 Pierre Fraigniaud, Magnús M. Halldórsson, and Alexandre Nolin. Distributed testing of distance- $k$  colorings. In *SIROCCO*, volume 12156 of *Lecture Notes in Computer Science*, pages 275–290. Springer, 2020.

- 12 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In *DISC*, volume 9888 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2016.
- 13 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.
- 14 Hiro Ito, Shin-Ichi Tanigawa, and Yuichi Yoshida. Constant-time algorithms for sparsity matroids. In *International Colloquium on Automata, Languages, and Programming*, pages 498–509. Springer, 2012.
- 15 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30. ACM/SIAM, 1993.
- 16 Reut Levi, Moti Medina, and Dana Ron. Property testing of planarity in the CONGEST model. *Distributed Comput.*, 34(1):15–32, 2021.
- 17 Y Orenstein. Testing properties of directed graphs. *Master’s thesis, School of Electrical Engineering*, 2010.
- 18 Merav Parter. Small cuts and connectivity certificates: A fault tolerant approach. In *DISC*, volume 146 of *LIPICs*, pages 30:1–30:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- 19 Shin-Ichi Tanigawa and Yuichi Yoshida. Testing the supermodular-cut condition. *Algorithmica*, 71(4):1065–1075, 2015.
- 20 Ramakrishna Thurimella. Sub-linear distributed algorithms for sparse certificates and biconnected components. *J. Algorithms*, 23(1):160–179, 1997.
- 21 Yuichi Yoshida and Hiro Ito. Testing  $k$ -edge-connectivity of digraphs. *Journal of systems science and complexity*, 23:91–101, 2010.
- 22 Yuichi Yoshida and Hiro Ito. Property testing on  $k$ -vertex-connectivity of graphs. *Algorithmica*, 62(3-4):701–712, 2012.

## A Robustness of Stochastic Closeness

The following claim is used to show robustness of stochastic closeness (Definition 4) with respect to the choice of the global constant  $c$ . Intuitively speaking, by repeating the random addition only a constant number of times, the probability of not attaining the property  $\mathcal{P}$  can be reduced to any small polynomial.

▷ **Claim 12.** Fix a monotone property  $\mathcal{P}$ , a graph  $G \notin \mathcal{P}$  over  $n$  vertices, and constant  $c > 1$ . If for some  $t \in [0, |\bar{E}|]$ , it holds that

$$\Pr[\text{Add}(G, t) \notin \mathcal{P}] \leq n^{-c},$$

then for any  $m \in \mathbb{N}$  such that  $mt \in [0, |\bar{E}|]$ , it holds that

$$\Pr[\text{Add}(G, mt) \notin \mathcal{P}] \leq n^{-mc}.$$

*Proof.* Recall that in the randomly augmented graph  $\text{Add}(G, t)$  each edge  $e \in \bar{E}$  is added independently with probability  $t/|\bar{E}|$ . Let  $G_1, \dots, G_m$  be  $m$  independent copies of  $\text{Add}(G, t)$ . By definition, we have

$$\Pr \left[ \bigwedge_{i=1}^m G_i \notin \mathcal{P} \right] = \prod_{i=1}^m \Pr[G_i \notin \mathcal{P}] \leq n^{-mc}.$$

Define  $G' = \bigcup_{i=1}^m G_i$ , the union of all  $m$  randomly augmented graphs.  $G'$  trivially contains all edges of  $G$ . On the one hand, the probability that each edge  $e \in \bar{E}$  appears in  $G'$  is at most  $(mt)/|\bar{E}|$ , by a union bound over the  $m$  attempts to add it. Since the property  $\mathcal{P}$  is monotone, we have

$$\Pr[\text{Add}(G, mt) \notin \mathcal{P}] \leq \Pr[G' \notin \mathcal{P}].$$


On the other hand,  $G'$  is a supergraph of  $G_i$  for all  $i \in [m]$ . Using again the monotonicity of  $\mathcal{P}$ , we can write

$$\Pr[G' \notin \mathcal{P}] = \Pr\left[\bigwedge_{i=1}^m G' \notin \mathcal{P}\right] \leq \Pr\left[\bigwedge_{i=1}^m G_i \notin \mathcal{P}\right] \leq n^{-mc},$$

which concludes the proof. ◁



# Expanderizing Higher Order Random Walks

Vedat Levi Alev  

Hebrew University of Jerusalem, Israel

Shravas Rao  

Portland State University, Portland, OR, United States of America

---

## Abstract

We study a variant of the down-up (also known as the Glauber dynamics) and up-down walks over an  $n$ -partite simplicial complex, which we call *expanderized higher order random walks* – where the sequence of updated coordinates correspond to the sequence of vertices visited by a random walk over an auxiliary expander graph  $H$ . When  $H$  is the clique with self loops on  $[n]$ , this random walk reduces to the usual down-up walk and when  $H$  is the directed cycle on  $[n]$ , this random walk reduces to the well-known systematic scan Glauber dynamics. We show that whenever the usual higher order random walks satisfy a log-Sobolev inequality or a Poincaré inequality, the expanderized walks satisfy the same inequalities with a loss of quality related to the two-sided expansion of the auxiliary graph  $H$ . Our construction can be thought as a higher order random walk generalization of the derandomized squaring algorithm of Rozenman and Vadhan (RANDOM 2005).

We study the mixing times of our expanderized walks in two example cases: We show that when initiated with an expander graph our expanderized random walks have mixing time (i)  $O(n \log n)$  for sampling a uniformly random list colorings of a graph  $G$  of maximum degree  $\Delta = O(1)$  where each vertex has at least  $(11/6 - \varepsilon)\Delta$  and at most  $O(\Delta)$  colors, (ii)  $O_{\mathbf{h}}\left(\frac{n \log n}{(1 - \|\mathbf{J}\|_{\text{op}})^2}\right)$  for sampling the Ising model with a PSD interaction matrix  $\mathbf{J} \in \mathbb{R}^{n \times n}$  satisfying  $\|\mathbf{J}\|_{\text{op}} \leq 1$  and the external field  $\mathbf{h} \in \mathbb{R}^n$  – here the  $O(\bullet)$  notation hides a constant that depends linearly on the largest entry of  $\mathbf{h}$ . As expander graphs can be very sparse, this decreases the amount of randomness required to simulate the down-up walks by a logarithmic factor.

We also prove some simple results which enable us to argue about log-Sobolev constants of higher order random walks and provide a simple and self-contained analysis of local-to-global  $\Phi$ -entropy contraction in simplicial complexes – giving simpler proofs for many pre-existing results.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Random walks and Markov chains; Theory of computation  $\rightarrow$  Expander graphs and randomness extractors; Theory of computation  $\rightarrow$  Generating random combinatorial structures

**Keywords and phrases** Higher Order Random Walks, Expander Graphs, Glauber Dynamics, Derandomized Squaring, High Dimensional Expansion, Spectral Independence, Entropic Independence

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.58

**Category** RANDOM

**Related Version** *Full version.*: <https://arxiv.org/abs/2405.08927> [4]

**Funding** *Vedat Levi Alev*: Supported by the ERC grant of Alex Lubotzky (European Union’s Horizon 2020/882751), the ISF grant 2669/21 and ERC grant 834735 of Gil Kalai, and ISF grant 2990/21 of Ori Parzanchevski.

*Shravas Rao*: This material is based upon work supported by the National Science Foundation under Award No. 2348489.

**Acknowledgements** We would like to thank Fernando Granha Jeronimo for many insightful discussions concerning expander graphs and anonymous referees for their many insightful comments.



© Vedat Levi Alev and Shravas Rao;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 58; pp. 58:1–58:24



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Let  $U_1, \dots, U_n$  be a collection of finite sets. The *down-up walk*  $P^{\downarrow\uparrow}$  on  $\Omega \subset U_1 \times \dots \times U_n$  with respect to a given distribution  $\pi : \Omega \rightarrow \mathbb{R}_{\geq 0}$ , also known as the *Glauber dynamics* on  $\Omega$  according to  $\pi$ , is the following simple process: Starting from an arbitrary tuple  $\omega^{(0)}$ , we obtain the  $(t+1)$ -st tuple  $\omega^{(t+1)}$  visited by this random walk from the  $t$ -th tuple  $\omega^{(t)}$  as follows,

### Update Rule for the Down-Up Walk, $P^{\downarrow\uparrow}$

1. sample a uniformly random coordinate  $i \sim \text{uni}_{[n]}$ ,
2. sample a random tuple  $\omega^{(t+1)} \sim \pi$  conditional on  $\omega_j^{(t+1)} = \omega_j^{(t)}$  for all  $j \in [n] \setminus \{i\}$ .

The following variant of the down-up walk, called the *systematic scan*  $P_{\text{scan}}$  on  $\Omega$  according to  $\pi$ , is a variant of the down-up walk  $P^{\downarrow\uparrow}$  which uses less randomness and is easier to implement in practice: starting from an arbitrary tuple  $\omega^{(0)}$ , we obtain the  $(t+1)$ -st tuple  $\omega^{(t+1)}$  visited by this random walk from the  $t$ -th tuple  $\omega^{(t)}$  as follows,

### Update Rule for the Systematic Scan, $P_{\text{scan}}$

1. set  $i = t + 1 \pmod{n}$ ,
2. sample a random tuple  $\omega^{(t+1)} \sim \pi$  conditional on  $\omega_j^{(t+1)} = \omega_j^{(t)}$  for all  $j \in [n] \setminus \{i\}$ .

In both cases, the coordinate  $i$  that is sampled on the first step of the update can be thought as a vertex visited by the simple random walk on a graph. For the down-up walk, this is a random walk on the clique with self-loops, whereas for the systematic scan this is a (deterministic) walk on the directed cycle.

The main object of study in this paper will be the so-called *expanderized down-up walk*  $Q^{\downarrow\uparrow}$  on  $\Omega$  with respect to the distribution  $\pi : \Omega \rightarrow \mathbb{R}_{> 0}$  and the  $k$ -regular graph  $H = ([n], E)$  for some constant  $k$ . Starting this random-walk from an arbitrary coordinate  $i^{(0)} \in [n]$  and an arbitrary tuple  $\omega^{(0)}$ , we obtain the  $(t+1)$ -st coordinate  $i^{(t+1)}$  and tuple  $\omega^{(t+1)}$  according to the following update rule,

### Update Rule for the Expanderized Down-Up Walk $Q^{\downarrow\uparrow}$

1. sample a random neighbor  $s$  of  $i^{(t)}$  in  $H$ ,
2. sample a random tuple  $\omega^{(t+1)} \sim \pi$  conditional on  $\omega_j^{(t+1)} = \omega_j^{(t)}$  for all  $j \in [n] \setminus \{s\}$ ,
3. set  $i^{(t+1)}$  to be a random neighbor of  $s$  in  $H$ .

We notice that according to the above update rule when  $i^{(0)}$  is sampled uniformly at random and  $H$  equals the clique with self-loops on  $[n]$  the evolution of  $\omega^{(t)}$  is as dictated by the down-up walk  $P^{\downarrow\uparrow}$ . Similarly, when  $i^{(0)} = 1$  and  $H$  is the directed cycle,<sup>1</sup> the evolution of  $\omega^{(t)}$  is as dictated by the systematic scan  $P_{\text{scan}}$ .

<sup>1</sup> To be more precise,  $H$  needs to be a directed cycle of length  $2n$ , in which the vertices corresponding to the coordinates and *dummy* vertices are interleaved.

The main contribution of this paper is an analysis of the expanderized down-up walk assuming, (i) the graph  $H$  is a spectral expander<sup>2</sup> and (ii) the down-up walk  $P^{\downarrow\uparrow}$  satisfies some kind of isoperimetric inequality, e.g. a log-Sobolev inequality or a Poincaré inequality. Indeed our methods allow us to extend our results to all down-up and up-down walks.

**Motivation and Contributions.** The systematic scan  $P_{\text{scan}}$  is a random walk of great practical and theoretical interest. Yet, rapid mixing results for this walk are only known under restricted circumstances [18, 33, 23, 24, 51, 29] and it is very hard to directly relate the rapid mixing of  $P^{\downarrow\uparrow}$  to that of  $P_{\text{scan}}$ . A particularly useful framework for establishing rapid mixing for the down-up walk is the method of high-dimensional expansion, in particular the frameworks of spectral independence and entropic independence [2, 9, 16, 17, 14, 7, 6, 8] which led to many breakthrough results in the field of sampling algorithms.

In [3], an attempt was made to study the mixing of the systematic scan<sup>3</sup> using techniques of high-dimensional expansion – while their techniques allowed them to establish rapid mixing results for constant dimensional partite simplicial complexes, their result is too restrictive to take advantage of mixing results obtained through spectral independence or entropic independence. As a step towards directly being able to take advantage of the mixing results for  $P^{\downarrow\uparrow}$ , which could potentially be obtained through the high-dimensional expansion framework, we introduce our expanderized down-up walks  $Q^{\downarrow\uparrow}$ . As expander graphs have proven themselves very successful at approximating dense objects, we hope – and indeed also prove – that transferring mixing time bounds from the usual down-up walks to our expanderized walks to be an easier task than establishing mixing times for  $P_{\text{scan}}$ . As expander graphs can be very sparse, our expanderized walks can be thought as replacing the sparse object used in the definition of the systematic scan  $P_{\text{scan}}$ , i.e. the directed cycle, with another sparse yet highly connected object – an expander graph with constant degree.

In spirit, the expanderized walks can be thought as a higher order random walk analogue of the derandomized squaring algorithm introduced in [52]. This algorithm was introduced to simplify the seminal result of [49] concerning the existence of a logspace algorithm for deciding undirected connectivity. The derandomized squaring operation uses an auxiliary  $k$ -regular expander graph  $H$  on the vertex set  $[d]$  to approximate the square of a graph  $d$ -regular graph  $G$  on  $[n]$ . Whereas the actual square  $G^2$  is a  $d^2$ -regular graph, by picking  $k = O(1)$  one can ensure that the *derandomized square* is  $O(d)$ -regular, i.e. a much sparser object. This result rests on the observation that the actual square  $G^2$  is obtained from the graph  $G$  by attaching a clique to every vertex – replacing this clique with an expander graph suffices to ensure that the resulting *derandomized square* is closed to the actual square. Fortunately, the same intuition also leads to proofs showing that the expanderized walks approximate the standard walks well.

We show that we can use our expanderized walks to have more randomness efficient versions of several Markov chains of interest for sampling list colorings [10, 41] and for sampling from Ising Models [27, 7, 40], under assumptions ensuring bounded marginals. In these settings, our expanderized walks have the same asymptotic mixing time but use fewer random bits. To help us with our goals we also prove some simple estimates for the log-Sobolev constants of higher order random walks and provide a self-contained analysis of local-to-global  $\Phi$ -entropy contraction.

<sup>2</sup> i.e. all non-trivial eigenvalues of  $H$  are bounded away from 1

<sup>3</sup> More formally,  $n$  successive steps of the systematic scan, which the authors call the *sequential sweep*  $P_{\text{seq}}$ .

**Related Work.** High dimensional expansion has proven itself to be a very successful research program for establishing mixing times for down-up walks. For example [36, 21, 37, 20, 2] use spectral local-to-global arguments for establishing spectral gap bounds for these walks. In conjunction with the spectral independence framework, due to [9, 15, 30], these results paved the way for many new in the field of random sampling: rapid mixing of the down-up walk for the hardcore model in the uniqueness regime [9], rapid mixing of the down-up walk for sampling graph colorings in correlation decay regime [30, 15], optimal mixing for many Markov chains of interest [17, 10, 41]. For more information regarding spectral independence, we refer the reader to the excellent survey [54] and the dissertation [42]. In [6, 8, 17, 32] local-to-global strategies for establishing entropic contraction bounds was studied. In [14] a connection between these local-to-global methods and the stochastic localization framework of [25] was explored. We refer to the works [39, 26, 13, 38, 28, 27] and references therein for applications of the stochastic localization framework. Our inductive strategy for establishing  $\Phi$ -entropy contraction on simplicial complexes is heavily inspired by the presentation in [14]. In [40] mixing estimates about the walk  $P_{n \leftrightarrow n-1}^{\downarrow \uparrow}$  is used to obtain estimates for  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  for all  $\ell < n - 1$ . The key intuition behind this work is the observation that the down move of the down-up walk is (passively) utilizing an expander, the down-move of the down-up walk of the so-called *Bernoulli-Laplace model*, and that one can use the expansion of this walk to show that once  $\ell$  decreases the mixing times estimates get better and better. Morally, this is very similar to our idea of picking the replacement-indices for our expanderized walks via an expander walk as opposed to sampling them uniformly at random. For other classical techniques which can be used to bound mixing times of Markov chains, we refer the reader to the texts [1, 46, 55].

In contrast with down-up walks, results establishing rapid mixing for the random walk  $P_{\text{scan}}$  are fewer [18, 33, 23, 51] and mostly rely on estimates on the Dobrushin matrix [22]. [3] studied the mixing time of this random walk using techniques of high dimensional expansion, however their techniques fell short of establishing mixing time bounds under the assumption of spectral independence.

The work of [29] is also related to our work in spirit. In this work, the authors show that under suitable assumptions a wide array of random walks, including the single site systematic scan  $P_{\text{scan}}$  and the down-up walk  $P^{\downarrow \uparrow}$ , can be derandomized, i.e. they devise efficient deterministic counting algorithms on the basis of rapid mixing results for these chains. It is an interesting question whether one can carefully pick the expander graph  $H$ , to make this derandomization task more efficient.

As mentioned above our expanderized random walks are heavily inspired by the derandomized squaring algorithm of [52]. This algorithm was initially used to give an alternative and simpler proof of the seminal result of [49] concerning the derandomization of the complexity class  $\mathbf{SL}$  and establishing  $\mathbf{SL} = \mathbf{L}$ . Concretely, both [49] and the subsequent work of [52] show the existence of a deterministic logspace algorithm deciding undirected graph connectivity. Since then, the derandomized squaring algorithm has also found other uses in derandomization, e.g. [47, 48]. We conclude by noting that the initial algorithm of [49] was based on the zigzag product construction [50], which has also inspired research in the field of high dimensional expansion [35]. For more information on expander graphs, we refer the reader to the excellent survey [34].



## 2 Preliminaries

### 2.1 Linear Algebra

We will denote functions and vectors by bold faces, i.e.  $\mathbf{f} \in \mathbb{R}^V$ . The indicator function of  $i \in V$  will be denoted by  $\mathbf{1}_i$ , i.e.  $\mathbf{1}_i(j) = 0$  for all  $j \neq i$  and  $\mathbf{1}_i(i) = 1$ . For  $A \subseteq V$ , we will write  $\mathbf{1}_A = \sum_{a \in A} \mathbf{1}_a$ . We will adopt the convention of using  $\pi, \nu, \mu : V \rightarrow \mathbb{R}_{\geq 0}$  for various probability distributions over  $V$ .

Let  $\mathbf{f}, \mathbf{g} \in \mathbb{R}^V$  and a measure  $\pi : V \rightarrow \mathbb{R}_{>0}$  be given. We will use the notations  $\langle \mathbf{f}, \mathbf{g} \rangle_\pi$  and  $\|\mathbf{f}\|_\pi$  to denote the inner-product and the norm with respect to the distribution  $\pi$ , i.e.

$$\langle \mathbf{f}, \mathbf{g} \rangle_\pi = \mathbb{E}_{x \sim \pi} \mathbf{f}(x)\mathbf{g}(x) = \sum_{x \in V} \pi(x) \cdot \mathbf{f}(x)\mathbf{g}(x) \quad \text{and} \quad \|\mathbf{f}\|_\pi^2 = \langle \mathbf{f}, \mathbf{f} \rangle_\pi. \quad (1)$$

Given  $\mathbf{f}, \mathbf{g} \in \mathbb{R}^n$  we will write  $\langle \mathbf{f}, \mathbf{g} \rangle_{\ell_2}$  for the inner-product between  $\mathbf{f}$  and  $\mathbf{g}$  in the counting measure, i.e.  $\langle \mathbf{f}, \mathbf{g} \rangle_{\ell_2} = \sum_{i=1}^n \mathbf{f}(i)\mathbf{g}(i)$ . We will also write  $\|\mathbf{f}\|_{\ell_1}$ ,  $\|\mathbf{f}\|_{\ell_2}$ , and  $\|\mathbf{f}\|_{\ell_\infty}$  for the  $\ell_1$ ,  $\ell_2$ , and  $\ell_\infty$  norms of  $\mathbf{f}$  respectively. Formally,

$$\|\mathbf{f}\|_{\ell_2}^2 = \sum_{i=1}^n \mathbf{f}(i)^2 \quad ; \quad \|\mathbf{f}\|_{\ell_1} = \sum_{i=1}^n |\mathbf{f}(i)| \quad ; \quad \text{and} \quad ; \quad \|\mathbf{f}\|_{\ell_\infty} = \max_{i \in [n]} |\mathbf{f}(i)|.$$

### Matrices and Eigenvalues

In this section, we will recall some results concerning eigenvalues and eigenvectors of matrices.

Serif faces will be used to denote matrices, i.e.  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{U \times V}$ . We will call a matrix  $\mathbf{B} \in \mathbb{R}^{U \times V}$  row stochastic if rows of  $\mathbf{B}$  sum up to 1 and  $\mathbf{B}$  contains no negative entries. Formally,

$$\text{for all } u \in U, v \in V \quad \mathbf{B}(u, v) \geq 0 \quad \text{and} \quad \mathbf{B}\mathbf{1} = \mathbf{1}. \quad (\text{row stochastic})$$

Let  $\mathbf{B} \in \mathbb{R}^{U \times V}$  and distributions  $\pi_U : U \rightarrow \mathbb{R}_{>0}$  and  $\pi_V : V \rightarrow \mathbb{R}_{>0}$  be given. The adjoint  $\mathbf{B}^*$  of  $\mathbf{B}$  with respect to the measures  $\pi_U$  and  $\pi_V$  is the unique matrix which satisfies the following equation,

$$\langle \mathbf{f}, \mathbf{B}\mathbf{g} \rangle_{\pi_U} = \langle \mathbf{B}^*\mathbf{f}, \mathbf{g} \rangle_{\pi_V} \quad \text{for all } \mathbf{f} \in \mathbb{R}^U, \mathbf{g} \in \mathbb{R}^V. \quad (\text{adjoint})$$

If  $U = V$  and  $\pi_U = \pi_V$ , the operator  $\mathbf{B}$  is called self-adjoint when  $\mathbf{B}^* = \mathbf{B}$ . If  $\mathbf{B}$  is a row-stochastic matrix, we will call  $\mathbf{B}^*$  the time-reversal of  $\mathbf{B}$  with respect to  $\pi_U, \pi_V$  and say that  $\mathbf{B}$  is reversible if  $\mathbf{B} = \mathbf{B}^*$ . It is well known that the operator  $\mathbf{B}^* \in \mathbb{R}^{V \times U}$  is uniquely determined by the choice of  $\mathbf{B} \in \mathbb{R}^{U \times V}$  and the inner-products defined by  $\pi_U$  and  $\pi_V$  (see e.g. [53, p. 318]),

► **Proposition 1.** *Let  $\mathbf{B} \in \mathbb{R}^{U \times V}$  be arbitrary. We write  $\mathbf{B}^*$  for the adjoint operator to  $\mathbf{B}$  with respect to the inner-products defined by the distributions  $\pi_U$  and  $\pi_V$ . Then,*

$$\mathbf{B}^*(y, x) = \mathbf{B}(x, y) \cdot \frac{\pi_U(x)}{\pi_V(y)} \quad \text{for all } x \in U, y \in V.$$

We also recall the following standard fact which is an immediate consequence of Proposition 1,

► **Proposition 2.** *If  $\mathbf{B} \in \mathbb{R}^{U, V}$  is a row-stochastic matrix satisfying  $\pi_U \mathbf{B} = \pi_V$ , then the adjoint matrix  $\mathbf{B}^*$  with respect to  $\pi_U, \pi_V$  is also row-stochastic and satisfies  $\pi_V \mathbf{B}^* = \pi_U$ .*

## 58:6 Expanderizing Higher Order Random Walks

It is well known that a self-adjoint matrix  $A \in \mathbb{R}^{V \times V}$  has  $|V|$  real eigenvalues. We will write,  $\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_{|V|}(A) := \lambda_{\min}(A)$  for the sequence of eigenvalues of  $A$  sorted in decreasing order. We say that the matrix is positive semi-definite, henceforth PSD, if it is self-adjoint and satisfies  $\lambda_{\min}(A) \geq 0$ .

Given a matrix  $A \in \mathbb{R}^{V \times V}$  and a distribution  $\mu : V \rightarrow \mathbb{R}_{>0}$ , we will write  $\|A\|_{\text{op},\mu}$  for the operator norm of  $A$ , defined in the following manner

$$\|A\|_{\text{op},\mu} := \max \left\{ \frac{\|A\mathbf{f}\|_\mu}{\|\mathbf{f}\|_\mu} \mid \mathbf{f} \in \mathbb{R}^V \text{ and } \mathbf{f} \neq 0 \right\}. \quad (\text{operator norm})$$

If  $A$  is self-adjoint with respect to the measure  $\mu$ , we have  $\|A\|_{\text{op},\mu} = \max\{\lambda_1(A), |\lambda_{\min}(A)|\}$ .

Similarly when  $A \in \mathbb{R}^{V \times V}$  is a reversible row-stochastic matrix, with stationary measure  $\mu$ . We will write  $\lambda(A)$  for the two-sided expansion of  $A$ . Formally,

$$\lambda(A) = \max\{\lambda_2(A), |\lambda_{\min}(A)|\}. \quad (\text{two-sided expansion})$$

When  $A$  represents the simple random walk over an undirected graph  $H = (V, E)$ , i.e.

$$A(i, j) = \frac{\mathbf{1}[\{i, j\} \in E]}{\deg(i)} \quad \text{for all } i, j \in V,$$

we will simply write  $\lambda(H)$  instead of  $\lambda(A)$ . For convenience, we recall

► **Observation 3.** *Let  $H = (V, E)$  be a  $k$ -regular graph and suppose  $A$  represents the random walk over  $H$ . Then,  $\text{uni}_V A = \text{uni}_V$ , i.e. the uniform distribution on  $V$  is stationary for  $A$ .*

We note that there exist infinite families of graphs such that every graph  $H$  in the family has constant degree and  $\lambda(H)$  bounded above by a constant bounded above by 1 [43, 44]. In this paper, we will consider families that contain graphs on  $n$  vertices for *every* sufficiently large  $n$ . Such constructions were given in [5], and in particular were based on the infinite families from [43, 44]. We refer the reader to the excellent survey [34] for more information on expander graphs.

We will also make use of the following simple result,

► **Lemma 4.** *Let a matrix  $A \in \mathbb{R}^{U \times V}$  and measures  $\mu_U : U \rightarrow \mathbb{R}_{>0}$  and  $\mu_V : V \rightarrow \mathbb{R}_{>0}$  be given, such that  $\mu_U A = \mu_V$ . Assume without loss of generality that  $|U| \leq |V|$ , then  $\lambda_j(AA^*) = \lambda_j(A^*A)$  for all  $j = 1, \dots, |U|$ , where  $A^*$  is the adjoint of  $A$  with respect to the measures  $\mu_U$  and  $\mu_V$ .*

## 2.2 Probability Distributions

Throughout the paper, we will assume  $\Omega$  (or  $X^{(n)}$ ) to be a set of  $n$ -tuples for some  $n \geq 1$ . Given a set  $S \subset [n]$ , the projection of  $\Omega$  on  $S$  is denoted by  $\Omega[S]$ , i.e.

$$\Omega[S] = \{(\omega_s)_{s \in S} : (\omega_1, \dots, \omega_n) \in \Omega\}. \quad (\text{projection})$$

Let  $\mu : \Omega \rightarrow \mathbb{R}_{\geq 0}$  be a distribution. For  $\omega_S \in \Omega[S]$ , the notations  $\Omega_{\omega_S}$  and  $\mu^{(\omega_S)}$  will be used for the  $\omega_S$ -pinning of  $\Omega$  and  $\mu_S$  respectively, where

$$\Omega_{\omega_S} = \{\bar{\omega} \in \Omega[S^c] : \omega_S \oplus \bar{\omega} \in \Omega\} \quad \text{and} \quad \mu^{(\omega_S)}(\bar{\omega}) = \frac{\mu(\omega_S \oplus \bar{\omega})}{\sum_{\tilde{\omega} \in \Omega[S^c]} \mu(\omega_S \oplus \tilde{\omega})}, \quad (\omega_S\text{-pinning})$$

We recall that the total variation distance  $\|\mu - \nu\|_{\text{tv}}$  between two distributions  $\mu, \nu : \Omega \rightarrow \mathbb{R}_{\geq 0}$  is defined as follows,

$$\|\mu - \nu\|_{\text{tv}} = \frac{1}{2} \cdot \sum_{\omega \in \Omega} |\mu(\omega) - \nu(\omega)| \quad (\text{total variation distance})$$

Finally, we talk about some conventions that we will use throughout the paper: (i) We will be using the notation  $\text{uni}_A$  to denote the uniform distribution over various finite sets  $A$ . (ii) When we want to emphasize that the a distribution  $\mu : \Omega \rightarrow \mathbb{R}_{\geq 0}$  has full support, we will simply write  $\mu : \Omega \rightarrow \mathbb{R}_{> 0}$ .

Finally we recall that the product distribution  $\mu \otimes \nu \in \Delta_{\Omega \times \Omega'}$ , given  $\mu \in \Delta_{\Omega}$  and  $\nu \in \Delta_{\Omega'}$  is defined by:  $(\mu \otimes \nu)(\omega, \omega') = \mu(\omega) \cdot \nu(\omega')$  for all  $\omega \in \Omega, \omega' \in \Omega'$ .

### 2.3 Functional Inequalities, Isoperimetric Constants, and Mixing Times

Given a distribution  $\mu \in \Delta_{\Omega}$  and a convex function  $\Phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$   $\Phi$ -entropy functional  $\text{Ent}_{\mu}^{\Phi}(\bullet)$  is defined by the equation,

$$\text{Ent}_{\mu}^{\Phi}(\mathbf{f}) = \mathbb{E}_{\omega \sim \mu} \Phi(\mathbf{f}(\omega)) - \Phi\left(\mathbb{E}_{\omega \sim \mu} \mathbf{f}(\omega)\right) \quad \text{for all } \mathbf{f} \in \mathbb{R}_{\geq 0}^{\Omega} \quad (\Phi\text{-entropy})$$

We also recall that for the special choices of  $\Phi(t) = t \log t$  and  $\Phi(t) = t^2$ , the  $\Phi$ -entropy equals the variance functional  $\text{Var}_{\mu}(\bullet)$  and entropy functional  $\text{Ent}_{\mu}(\bullet)$  respectively.

$$\text{Ent}_{\mu}(\mathbf{f}) = \mathbb{E}_{\omega \sim \mu} [\mathbf{f}(\omega) \log \mathbf{f}(\omega)] - \left(\mathbb{E}_{\omega \sim \mu} \mathbf{f}(\omega)\right) \log\left(\mathbb{E}_{\omega \sim \mu} \mathbf{f}(\omega)\right), \quad (\text{entropy})$$

$$\text{Var}_{\mu}(\mathbf{f}) = \mathbb{E}_{\omega \sim \mu} \mathbf{f}(\omega)^2 - \left(\mathbb{E}_{\omega \sim \mu} \mathbf{f}(\omega)\right)^2. \quad (\text{variance})$$

Let  $\mathbf{P} \in \mathbb{R}^{\Omega \times \Omega}$  be a reversible Markov chain, with stationary measure of  $\pi$ . A Poincaré inequality for  $\mathbf{P}$  is an inequality of the form,

$$C \cdot \text{Var}_{\pi}(\mathbf{f}) \leq \langle \mathbf{f}, (1 - \mathbf{P})\mathbf{f} \rangle_{\pi} \quad \text{for all } \mathbf{f} \in \mathbb{R}^{\Omega}. \quad (\text{Poincaré inequality})$$

The largest constant  $C > 0$  for which this inequality holds, is called the Poincaré constant or the spectral gap of  $\mathbf{P}$  and is denoted by  $\text{gap}(\mathbf{P})$ . This nomenclature is due to the following well-known consequence of the Courant-Fischer-Weyl Principle,

$$\text{gap}(\mathbf{P}) = \min \left\{ \frac{\langle \mathbf{f}, (1 - \mathbf{P})\mathbf{f} \rangle_{\pi}}{\text{Var}_{\pi}(\mathbf{f})} \mid \text{Var}_{\pi}(\mathbf{f}) \neq 0 \right\} = 1 - \lambda_2(\mathbf{P}). \quad (\text{spectral gap})$$

The log-Sobolev (LSI) inequality for a reversible random walk  $\mathbf{P} \in \mathbb{R}^{\Omega \times \Omega}$  with stationary measure  $\pi$  is defined to be,

$$C \cdot \text{Ent}_{\pi}(\mathbf{f}^2) \leq \langle \mathbf{f}, (1 - \mathbf{P})\mathbf{f} \rangle_{\pi} \quad \text{for all } \mathbf{f} \in \mathbb{R}_{\geq 0}^{\Omega}. \quad (\text{LSI})$$

The largest constants  $C \geq 0$  for which LSI holds is called the log-Sobolev constant of  $\mathbf{P}$  respectively and is denoted by  $\text{ls}(\mathbf{P})$ . Formally,

$$\text{ls}(\mathbf{P}) = \inf \left\{ \frac{\langle \mathbf{f}, (1 - \mathbf{P})\mathbf{f} \rangle_{\pi}}{\text{Ent}_{\pi}(\mathbf{f}^2)} \mid \text{Ent}_{\pi}(\mathbf{f}) \neq 0, \mathbf{f} \in \mathbb{R}_{\geq 0}^{\Omega} \right\}. \quad (2)$$

► **Lemma 5** ([19]). *Let  $\pi : \Omega \rightarrow \mathbb{R}_{>0}$  be a probability distribution and write  $J_\pi = \mathbf{1} \cdot \pi$ , i.e.  $J_\pi$  is the walk with stationary measure  $\pi$  which mixes in a single step.*

*Then,  $\mathbf{1s}(J_\pi) \geq \frac{1-2\pi_*}{\log(\pi_*^{-1}-1)}$  if  $|\text{supp}(\pi)| > 2$  else  $\mathbf{1s}(J_\pi) = 1$ . More generally for any reversible Markov chain  $M \in \mathbb{R}^{\Omega \times \Omega}$  and stationary distribution  $\pi$ , we have  $\mathbf{1s}(M) \geq \frac{1-2\pi_*}{\log(\pi_*^{-1}-1)} \cdot \text{gap}(M)$  if  $|\text{supp}(\pi)| > 2$  else  $\mathbf{1s}(J_\pi) = \text{gap}(M)$ .*

For a convex function  $\Phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ , we also define the  $\Phi$ -entropy contraction constant  $\text{cf}_\Phi(P)$  of a Markov chain  $P \in \mathbb{R}^{\Omega_1 \times \Omega_2}$  satisfying  $\pi_1 P = \pi_2$  for some choice of measures  $\pi_1 \in \Delta_{\Omega_1}, \pi_2 \in \Delta_{\Omega_2}$ , as the solution to the following variational problem,

$$\text{cf}_\Phi(P) = 1 - \sup \left\{ \frac{\text{Ent}_{\pi_1}^\Phi(P\mathbf{f})}{\text{Ent}_{\pi_2}^\Phi(\mathbf{f})} \mid \mathbf{f} \in \mathbb{R}_{\geq 0}^\Omega, \text{Ent}_{\pi_2}^\Phi(\mathbf{f}) \neq 0 \right\}. \quad (\Phi\text{-entropy contraction})$$

We note that  $\text{cf}_\Phi(P)$  crucially depends on the choice of distributions  $\pi_1, \pi_2$ . Since for our purposes the choice of measures  $\pi_1$  and  $\pi_2$  will always be clear, we will suppress this dependency.

It is equivalent to define  $\text{cf}_\Phi(P)$  as the largest constant  $C \in \mathbb{R}_{\geq 0}$  such that the inequality,

$$\text{Ent}_{\pi_1}^\Phi(P\mathbf{f}) \leq (1 - C) \cdot \text{Ent}_{\pi_2}^\Phi(\mathbf{f}),$$

is valid for each  $\mathbf{f} \in \mathbb{R}_{\geq 0}^{\Omega_2}$ . When  $\Phi(t) = t \log t$ , we will simply write  $\text{ec}(P)$  in place of  $\text{cf}_\Phi(P)$ . Similarly, for the choice of  $\Phi(t) = t^2$ , it is easy to observe that  $\text{cf}_\Phi(P) = \text{gap}(P^*P)$ .

We will also need the following consequence of Jensen's inequality.

► **Lemma 6** (Data Processing Inequality). *Let  $P \in \mathbb{R}^{\Omega_1 \times \Omega_2}$  be a row-stochastic matrix, satisfying  $\pi_1 P = \pi_2$  for probability distributions  $\pi_1 : \Omega_1 \rightarrow \mathbb{R}_{>0}$  and  $\pi_2 : \Omega_2 \rightarrow \mathbb{R}_{>0}$ . Then, for any convex function  $\Phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ , we have:  $\text{Ent}_{\pi_1}^\Phi(P\mathbf{f}) \leq \text{Ent}_{\pi_2}^\Phi(\mathbf{f})$  for all  $\mathbf{f} \in \mathbb{R}_{\geq 0}^{\Omega_2}$ .*

► **Lemma 7** (Proposition 6, [45]). *Let  $P \in \mathbb{R}^{\Omega_1 \times \Omega_2}$  satisfying  $\mu_1 P = \mu_2$ , for distributions  $\mu_1 : \Omega_1 \rightarrow \mathbb{R}_{>0}$  and  $\mu_2 : \Omega \rightarrow \mathbb{R}_{>0}$ . We have,  $\text{ec}(P) \geq \mathbf{1s}(P^*P)$ .*

The  $\varepsilon$ -mixing time  $\tau_{\text{mix}}(P, \varepsilon)$  of the random walk is the least time point  $t \in \mathbb{N}$ , such that the distribution  $\mu^{(t)} = \mu^{(0)} P^t$  of the random walk  $P$  is guaranteed to be  $\varepsilon$ -close to the stationary distribution  $\pi$  in the total variation distance regardless of the initial distribution  $\mu^{(0)}$ . In particular,

$$\tau_{\text{mix}}(P, \varepsilon) = \min \left\{ t \in \mathbb{N} \mid \left\| \mu^{(t)} - \pi \right\|_{\text{tv}} \leq \varepsilon \text{ for all } \mu^{(0)} \in \Delta_\Omega \right\} \quad (\varepsilon\text{-mixing time})$$

It is well known that the functional inequalities and the corresponding isoperimetric constants introduced previously can be used to bound mixing times. We recall in particular,

► **Theorem 8** ([11]). *There exists a universal constant  $C$  such that, for any reversible random walk  $P \in \mathbb{R}^{\Omega \times \Omega}$  with stationary distribution  $\pi : \Omega \rightarrow \mathbb{R}_{>0}$ , i.e.  $\pi P = \pi$ . We have*

$$\tau_{\text{mix}}(P, \varepsilon) \leq \frac{C}{\text{ec}(P)} \cdot \left( \log \log \frac{1}{\min_{\omega \in \Omega} \pi(\omega)} + \log \varepsilon^{-1} \right).$$

where the constant  $C$  does not depend on the pair  $(P, \pi)$ .

## 2.4 (Partite) Simplicial Complexes

A simplicial complex is a downward closed collection of subsets of a finite set  $U$ . Formally,  $X \subset 2^U$  and whenever  $\beta \in X$  for all  $\alpha \subset \beta$  we have  $\alpha \in X$ . The rank of a face  $\alpha$  is  $|\alpha|$ . Given some  $j$ , we will adopt the notation  $X^{(j)}$  to refer to the collection faces of  $X$  of rank  $j$

and the notation  $X^{(\leq j)}$  to refer to the collection of faces of  $X$  of rank at most  $j$ . We say  $X$  is a simplicial complex of rank  $n$  if the largest rank of any face  $\alpha \in X$  is  $n$ . We note that by definition  $X^{(0)} = \{\emptyset\}$ .

We say that a simplicial complex  $X$  of rank  $n$  is pure, if any face  $\alpha \in X^{(j)}$  for any  $j < n$  is contained in another face  $\beta \in X^{(n)}$ . Equivalently, in a pure simplicial complex the only inclusion maximal faces are those of maximal rank. In this article, we will only deal with pure simplicial complexes.

A rank- $n$  pure simplicial complex  $X$  is called  $n$ -partite if we can partition  $X^{(1)}$  into disjoint sets  $X[1], \dots, X[n]$  such that

$$\text{for all } \beta \in X^{(n)} \text{ and for all } i = 1, \dots, n \text{ we have } |\beta \cap X[i]| = 1. \quad (n\text{-partiteness})$$

We will call the sets  $X[1], \dots, X[n]$  the sides of the complex  $X$ . Equivalently, every element of a rank- $n$  face  $\beta \in X[n]$  comes from a distinct side  $X[i]$ . We observe that a bipartite graph is a 2-partite simplicial complex.

To keep our nomenclature simple, we will simply refer to a pure  $n$ -partite simplicial complex of rank  $n$  as an  $n$ -partite simplicial complex, i.e. we will not consider  $n$ -partite complexes which are not pure.

For a face  $\alpha \in X$  we introduce the notation,  $\text{type}(\alpha) = \{i \in [n] : \alpha \cap X[i] \neq \emptyset\}$  for the type of the face  $\alpha$ , i.e. the collection of sides of  $X$  that  $\alpha$  intersects.

For any  $i \in [n]$  and  $\beta \in X^{(n)}$  we will write  $\beta_i \in X^{(1)}$  for the unique element of  $\beta$  satisfying  $\{\beta_i\} = \beta \cap X[i]$ . We will refer to  $\beta_i$  as the  $i$ -th coordinate of  $\beta$ . We will also write  $\beta_T = \{\beta_t : t \in T\}$  for all  $T \subset [n]$ . We extend this notation to arbitrary faces  $\alpha \in X$  and  $T \subset \text{type}(\alpha)$ . In keeping with the view that a face  $\alpha \in X$  with  $\text{type}(\alpha) = \{t_1, \dots, t_k\}$  can be represented as a tuple  $(a_{t_1}, \dots, a_{t_k})$ , we will favour the notation  $\alpha \oplus \alpha'$  to denote the union of two faces  $\alpha, \alpha' \in X$  with  $\text{type}(\alpha) \cap \text{type}(\alpha') = \emptyset$  over the usual notation  $\alpha \cup \alpha'$ .

We observe that for facets  $\beta \in X^{(n)}$ , i.e. faces of maximal rank, we have  $\text{type}(\beta) = [n]$ . Given,  $\alpha \in X$  we recall that the link  $X_\alpha$  is defined as,  $X_\alpha = \{(\beta \setminus \alpha) \in X : \beta \in X, \beta \supset \alpha\}$ .

For  $T \subset [n]$ , we will also introduce the notation  $X[T]$  to refer to all faces of  $X$  of type  $T$ .

### Weighted Simplicial Complexes

A weighted simplicial complex  $(X, \pi)$  of rank  $n$  is a pure simplicial complex of rank  $n$  where  $\pi := \pi_n : X^{(n)} \rightarrow \mathbb{R}_{\geq 0}$  is a probability distribution with full support.

For  $j \in [0, n - 1]$ , we inductively define the probability distributions  $\pi_j : X^{(j)} \rightarrow \mathbb{R}$  as

$$\pi_j(\alpha) = \pi_j(\alpha) = \frac{1}{\binom{n}{j}} \sum_{\substack{\beta \supset \alpha \\ \beta \in X^{(n)}}} \pi_n(\beta). \quad (3)$$

Similarly, given a face  $\alpha \in X^{(j)}$ , we define the distribution  $\pi^{(\alpha)}$  on  $X_\alpha^{(n-j)}$  by conditioning  $\pi$  on the containment of  $\alpha$ . Of particular importance to us will be the link graph  $M_\alpha \in \mathbb{R}^{X_\alpha^{(1)} \times X_\alpha^{(1)}}$  given any  $\alpha \in X^{(\leq n-2)}$ . We recall that for all distinct pairs of vertices  $x, y \in X_\alpha^{(1)}$ , we have

$$M_\alpha(x, y) = \pi^{(\alpha \cup \{x\})}(y) = \frac{\Pr_{\omega \sim \pi_n}[\omega \supset \alpha \cup \{x, y\} \mid \omega \supset \alpha \cup \{x\}]}{n - |\alpha| - 1}, \quad (\text{link})$$

and  $M_\alpha(x, x) = 0$  for all  $x \in X_\alpha^{(1)}$ .

## 2.5 Higher Order Random Walks on Simplicial Complexes

Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . The up-down walk  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi)$  between the  $\ell$ -th and  $n$ -th levels,  $X^{(\ell)}$  and  $X^{(n)}$  respectively, is defined as the following random walk on  $X^{(\ell)}$ : Starting from an arbitrary face  $\hat{\omega}^{(0)} \in X^{(\ell)}$  for all  $t \geq 1$  move from  $\hat{\omega}^{(t-1)}$  to  $\hat{\omega}^{(t)}$  according to the following simple rule,

**Update Rule For the Up-Down Walk,  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow}$**

- sample  $\omega \sim \pi_n$ , conditional on  $\omega \supset \hat{\omega}^{(t-1)}$ ,
- draw a uniformly subset among all the subsets of  $\omega$  of size  $\ell$ , and output it as  $\hat{\omega}$ .

Similarly, the down-up walk  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  between the  $n$ -th and  $\ell$ -th levels,  $X^{(n)}$  and  $X^{(\ell)}$  respectively, as the following random walk  $X^{(n)}$ : Starting from an arbitrary  $\omega^{(0)} \in X^{(n)}$  and moves from  $\omega^{(t-1)}$  to  $\omega^{(t)}$  according to the following simple rule,

**Update Rule for the Down-Up Walk,  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$**

- draw a subset  $\hat{\omega}$  of  $\omega$  of size  $\ell$ , uniformly at random,
- draw a subset  $\omega \sim \pi$  conditioned on containing  $\hat{\omega}$ , and output it as  $\omega^{(t)}$ .

It is well known, [2, 21, 20], that the random walks  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  and  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  can be decomposed as a product of random down- and up-movements on  $X$ . Formally, for  $0 \leq \ell \leq k \leq n$ , we define the up-walk  $P_{\ell \rightarrow k}^{\uparrow} := \text{Up}_{\ell \rightarrow k}(X, \pi)$  and the down-walk  $P_{k \rightarrow \ell}^{\downarrow} := \text{Down}_{k \rightarrow \ell}(X, \pi)$  as the following random walks,

$$P_{\ell \rightarrow k}^{\uparrow}(\hat{\omega}, \omega) = \pi_{k-\ell}^{(\hat{\omega})}(\omega) = \frac{\mathbf{1}[\omega \supset \hat{\omega}] \cdot \Pr_{\omega \sim \pi_n}[\tilde{\omega} \supset \omega \mid \tilde{\omega} \supset \hat{\omega}]}{\binom{n-\ell}{k-\ell}}, \quad (\text{up-walk})$$

$$P_{k \rightarrow \ell}^{\downarrow}(\omega, \hat{\omega}) = \frac{\mathbf{1}[\hat{\omega} \subset \omega]}{\binom{k}{\ell}}. \quad (\text{down-walk})$$

► **Proposition 9 (Folklore).** *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ , then writing  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow} := \text{DownUp}_{n \leftrightarrow \ell}(X, \pi)$ ,  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} := P_{\ell \leftrightarrow n}^{\uparrow \downarrow}(X, \pi)$ ,  $P_{\ell \rightarrow n}^{\uparrow} = \text{Up}_{\ell \rightarrow n}(X, \pi)$ , and  $P_{n \rightarrow \ell}^{\downarrow} = \text{Down}_{n \rightarrow \ell}(X, \pi)$  for the down-up, up-down, up- and down-walks between the  $n$ -th and  $\ell$ -th levels of  $X$  respectively, we have*

1.  $(P_{\ell \rightarrow n}^{\uparrow})^* = P_{n \rightarrow \ell}^{\downarrow}$ , i.e. the operators  $P_{\ell \rightarrow n}^{\uparrow}$  and  $P_{n \rightarrow \ell}^{\downarrow}$  are adjoint operators with respect to the measures  $\pi_n$  and  $\pi_\ell$ ,
2.  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} = P_{\ell \rightarrow n}^{\uparrow} P_{n \rightarrow \ell}^{\downarrow}$  - in particular the operator  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  is PSD,
3.  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow} = P_{n \rightarrow \ell}^{\downarrow} P_{\ell \rightarrow n}^{\uparrow}$  - in particular the operator  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  is PSD.

For any  $\hat{\omega} \in X$  and any  $0 \leq \ell \leq n' = n - |\hat{\omega}|$ , we will write  $P_{\hat{\omega}, \ell \rightarrow n'}^{\uparrow}$ ,  $P_{\hat{\omega}, n' \rightarrow \ell}^{\downarrow}$ ,  $P_{\hat{\omega}, \ell \leftrightarrow n'}^{\uparrow \downarrow}$  and  $P_{\hat{\omega}, n' \leftrightarrow \ell}^{\downarrow \uparrow}$  for the corresponding up, down, up-down, and down-up walks in the complex  $(X_{\hat{\omega}}, \pi_{\hat{\omega}})$ .

## 2.6 Local to Global Analysis

Given a simplicial complex  $(X, \pi)$  of rank  $n$ , we define the local  $\Phi$ -entropy contraction factor  $\text{lc}_\Phi(\hat{\omega})$  for any  $\hat{\omega} \in X^{(\leq r-2)}$  as follows,

$$\text{lc}_\Phi(\hat{\omega}) := \sup \left\{ \frac{\text{Ent}_{\pi_1^{(\hat{\omega})}}^\Phi(P_{\hat{\omega}, 1 \rightarrow n'}^{\uparrow} \mathbf{g})}{\text{Ent}_{\pi_{n'}^{(\hat{\omega})}}^\Phi(\mathbf{g})} \mid \mathbf{g} \in \mathbb{R}^{X_{\hat{\omega}}^{(n')}} \text{ and } n' = n - |\hat{\omega}|. \right\} \quad (4)$$

Equivalently,  $\mathbf{1c}_\Phi(\hat{\omega}) \in \mathbb{R}_{>0}$  is the smallest constant satisfying the equality

$$\text{Ent}_{\pi_1}^\Phi(\mathbf{P}_{\hat{\omega},1 \rightarrow n'}^\uparrow \mathbf{g}) \leq \mathbf{1c}_\Phi(\hat{\omega}) \cdot \text{Ent}_{\pi_{n'}}^\Phi(\mathbf{g}) \quad \text{for all } \mathbf{g} \in \mathbb{R}^{X_{\hat{\omega}}^{(n')}} \quad \text{where } n' = n - |\hat{\omega}|.$$

When  $\Phi(t) = t \log t$ , we will simply write  $\mathbf{1ec}(\hat{\omega})$  in place of  $\mathbf{1c}_\Phi(\hat{\omega})$ . We also make the following observation for the special case  $\Phi(t) = t^2$ , i.e. when  $\text{Ent}_{\Phi}^\bullet(\bullet)$  equals the variance functional  $\text{Var}_\bullet(\bullet)$ . The following proposition is well understood,

► **Proposition 10.** <sup>4</sup> *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . Then, for the choice of  $\Phi(t) = t^2$ , for any  $\hat{\omega} \in X^{(\leq n-2)}$  we have*

$$\mathbf{1c}_\Phi(\hat{\omega}) = \frac{1}{n - |\hat{\omega}|} + \frac{n - |\hat{\omega}| - 1}{n - |\hat{\omega}|} \cdot \lambda_2(\mathbf{M}_{\hat{\omega}}),$$

where  $\mathbf{M}_{\hat{\omega}}$  is the link graph of  $\hat{\omega}$ .

A crucial tool we will be using in Section 4 is the so called Garland method, due to [31]. To this end, we define the localization  $\mathbf{f}|_{\hat{\omega}} \in \mathbb{R}^{X_{\hat{\omega}}^{(k-j)}}$  of a function  $\mathbf{f} \in \mathbb{R}^{X^{(k)}}$  on a link  $\hat{\omega} \in X^{(j)}$  for  $j \leq k$  as the following function,

$$\mathbf{f}|_{\hat{\omega}}(\alpha) = \mathbf{f}(\hat{\omega} \sqcup \alpha) \quad \text{for all } \alpha \in X_{\hat{\omega}}^{(k-j)}. \quad (\text{localization})$$

We first observe that by appealing to the chain rule for the  $\Phi$ -entropy, one can obtain a convenient expression for it in terms of localizations.

► **Lemma 11** (Chain Rule for  $\Phi$ -Entropy). <sup>5</sup> *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . For all  $0 \leq \ell \leq r \leq n$  and non-negative  $\mathbf{f} \in \mathbb{R}_{\geq 0}^{X^{(r)}}$ , we have*

$$\text{Ent}_{\pi_r}^\Phi(\mathbf{f}) = \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \text{Ent}_{\pi_{r-\ell}}^\Phi(\mathbf{f}|_{\hat{\omega}}) + \text{Ent}_{\pi_\ell}(\mathbf{P}_{\ell \rightarrow r}^\uparrow \mathbf{f}),$$

where  $\mathbf{P}_{\ell \rightarrow r}^\uparrow := \text{Up}_{\ell \rightarrow r}(X, \pi)$  is the up-walk on  $X$

We also recall the following identities,

► **Lemma 12.** *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . Writing  $\mathbf{P}_{n \leftrightarrow r}^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow r}(X, \pi)$ ,  $\mathbf{P}_{n-1}^{\uparrow \downarrow} = \text{UpDown}_{n-1 \leftrightarrow n}(X, \pi)$ , and  $\mathbf{M}_{\hat{\omega}}$  for the link of the face  $\hat{\omega} \in X^{(\leq n-2)}$ , for all  $\mathbf{f} \in \mathbb{R}^{X^{(n)}}$  and  $\ell \leq r \leq n$ , we have*

1.  $\langle \mathbf{f}, \mathbf{f} \rangle_{\pi_n} = \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \langle \mathbf{f}|_{\hat{\omega}}, \mathbf{f}|_{\hat{\omega}} \rangle_{\pi_{n-\ell}^{(\hat{\omega})}},$
2.  $\langle \mathbf{f}, \mathbf{P}_{n \leftrightarrow r}^{\downarrow \uparrow} \mathbf{f} \rangle_{\pi_n} = \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \langle \mathbf{f}|_{\hat{\omega}}, \mathbf{P}_{\hat{\omega}, n-\ell \leftrightarrow r-\ell}^{\downarrow \uparrow} \mathbf{f}|_{\hat{\omega}} \rangle_{\pi_{n-\ell}^{(\hat{\omega})}},$
3.  $\langle \mathbf{f}, \mathbf{P}_{n-1}^{\uparrow \downarrow} \mathbf{f} \rangle_{\pi_n} = \mathbb{E}_{\hat{\omega} \sim \pi_{n-2}} \left( \langle \mathbf{f}|_{\hat{\omega}}, \left( \frac{1}{n} + \frac{n-1}{n} \cdot \mathbf{M}_{\hat{\omega}} \right) \mathbf{f}|_{\hat{\omega}} \rangle_{\pi_1^{(\hat{\omega})}} \right)$

We recall the following result due to [40],

► **Lemma 13** (Theorem 3.5, [40]). *Let  $(X, \pi)$  be an  $n$ -partite simplicial complex and let  $0 \leq k < n$ . If  $\mathbf{ec}(\mathbf{P}_{n-1 \rightarrow n}^\uparrow) \geq (Cn)^{-1}$  for some  $C \in \mathbb{R}_{>0}$ , then  $\mathbf{ec}(\mathbf{P}_{k \rightarrow k+1}^\uparrow) \geq \frac{1}{(k+1)(C+1)}$ .*

<sup>4</sup> We provide a proof for this statement in the full version of our paper, [4]

<sup>5</sup> A proof is supplied in the full version of our paper, [4]

### 3 Expanderized Random Walks

Let  $(X, \pi)$  be an  $n$ -partite simplicial complex. For any  $\ell \leq n$ , the up-down walk  $\mathbb{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi)$  on the  $\ell$ -th level  $X^{(\ell)}$  of  $X$  introduced in Section 2.5 admits the following alternative description: Starting from an arbitrary face  $\hat{\omega}^{(0)} \in X^{(\ell)}$  move from  $\hat{\omega}^{(t-1)}$  to  $\hat{\omega}^{(t)}$  according to the following simple rule,

**Update Rule for the Up-Down Walk,  $\mathbb{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$**

- sample  $\omega \sim \pi$ , conditional on  $\omega \supset \hat{\omega}^{(t-1)}$ ,
- sample  $S \sim \text{uni}_{\binom{[n]}{\ell}}$ ,
- output  $\hat{\omega}^{(t)} = \omega_S$ .

We will expanderize the up-down walk  $\mathbb{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  in the following manner: Given a  $k$ -regular labelled graph  $H$  on the vertex set  $\binom{[n]}{\ell}$ , we will denote the  $a$ -th neighbor of vertex  $v$  by  $\text{Out}_H(v, a)$ . We define the expanderized up-down walk  $\mathbb{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow} = \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H)$  as the walk which starts from an arbitrary face  $\hat{\omega}^{(0)}$  and moves from  $\hat{\omega}^{(t-1)}$  to  $\hat{\omega}^{(t)}$  according to the following simple rule,

**Update Rule for the Expanderized Up-Down Walk,  $\mathbb{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$**

- sample  $\omega \sim \pi$ , conditional on  $\omega \supset \hat{\omega}^{(t-1)}$ ,
- sample  $a \sim \text{uni}_{[k]}$  and set  $S = \text{Out}_H(\text{type}(\hat{\omega}^{(t-1)}), a)$ ,<sup>a</sup>
- output  $\hat{\omega}^{(t)} = \omega_S$ .

<sup>a</sup> Where we recall that the type of  $\hat{\omega}$  is the sides of the simplicial complex that  $\hat{\omega}$  intersects

Similarly, the down-up walk  $\mathbb{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  between the  $n$ -th level  $X^{(n)}$  and the  $\ell$ -th level  $X^{(\ell)}$  of an  $n$ -partite simplicial complex  $(X, \pi)$  introduced in Section 2.5 admits the following alternative description: Start from  $\omega^{(0)} \in X^{(n)}$  and move from  $\omega^{(t-1)}$  to  $\omega^{(t)}$  according to the following simple rule,

**Update Rule for the Down-Up Walk,  $\mathbb{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}$**

- sample  $S \sim \text{uni}_{\binom{[n]}{\ell}}$  uniformly at random,
- set  $\hat{\omega} = \omega_S$ ,
- set  $\omega^{(t)}$  to be a random face drawn from  $\pi$ , conditional on containing  $\hat{\omega}$ .

Similarly, we define the expanderized down-up walk  $\mathbb{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow \ell}(X, \pi, H)$  to be the random walk on  $X^{(n)} \times \binom{[n]}{\ell}$ , starting from an arbitrary face-subset pair  $(\omega^{(0)}, S^{(0)})$  and move from  $(\omega^{(t-1)}, S^{(t-1)})$  to  $(\omega^{(t)}, S^{(t)})$  according to the following simple rule,

**Update Rule for the Expanderized Down-Up Walk,  $\mathbb{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow}$**

- sample  $a \sim \text{uni}_{[k]}$  and set  $S' = \text{Out}_H(S^{(t-1)}, a)$ ,
- set  $\hat{\omega} = \omega_{S'}$ ,
- set  $\omega^{(t)} \sim \pi$  to be a random face conditional on containing  $\hat{\omega}$ ,
- sample  $b \sim \text{uni}_{[k]}$  and set  $S^{(t)} = \text{Out}_H(S^{(t-1)}, b)$ ,
- output  $(\omega^{(t)}, S^{(t)})$ .



For convenience we also define the expanderized down- and up-walks given a degree regular labelled graph  $H = (\binom{[n]}{\ell}, E)$

$$\mathbf{Q}_{n \rightarrow \ell}^\downarrow = \text{Down}_{n \rightarrow \ell}(X, \pi, H) \in \mathbb{R}^{(X^{(n)} \times \binom{[n]}{\ell}) \times X^{(\ell)}} \quad \text{and} \quad \mathbf{Q}_{\ell \rightarrow n}^\uparrow = \text{Up}_{\ell \rightarrow [n]}(X, \pi, H) \in \mathbb{R}^{X^{(\ell)} \times (X^{(n)} \times \binom{[n]}{\ell})},$$

as follows,

$$\mathbf{Q}_{n \rightarrow \ell}^\downarrow((\omega, S), \hat{\omega}) = \frac{\mathbf{1}[S \sim_H \text{type}(\hat{\omega})]}{k} \cdot \mathbf{1}[\omega \supset \hat{\omega}] \quad \text{for all } \omega \in X^{(n)}, \hat{\omega} \in X^{(\ell)}, S \in \binom{[n]}{\ell},$$

and  $\mathbf{Q}_{\ell \rightarrow n}^\uparrow = (\mathbf{Q}_{n \rightarrow \ell}^\downarrow)^*$  where the adjoint is taken with respect to the distributions  $\pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}$  and  $\pi_\ell$ , i.e.

$$\mathbf{Q}_{\ell \rightarrow n}^\uparrow(\hat{\omega}, (\omega, S)) = \frac{\mathbf{1}[S \sim_H \text{type}(\hat{\omega})]}{k} \cdot \Pr_{\tilde{\omega} \sim \pi_n} [\tilde{\omega} = \omega \mid \omega \supset \hat{\omega}] \quad \text{for all } \omega \in X^{(n)}, \hat{\omega} \in X^{(\ell)}, S \in \binom{[n]}{\ell},$$

and the notation  $T \sim_H S$  is used to denote the adjacency relation in the graph  $H$ , i.e.  $\{S, T\} \in E(H)$ .

We summarize the random movements described by the expanderized up- and down-walks in words as follows: The expanderized down-walk  $\mathbf{Q}_{n \rightarrow \ell}^\downarrow$  first samples a random neighbor of  $T$  of  $S$  in  $\binom{[n]}{\ell}$ , and then restricts the coordinates of  $\omega$  to  $T$ , i.e. moves to  $\omega_T$ . The expanderized up-walk  $\mathbf{Q}_{\ell \rightarrow n}^\uparrow$  on the other hand first samples a facet  $\omega \in X^{(n)}$  from  $\pi$  conditional on containing  $\hat{\omega}$  and after picking a random neighbor  $S$  of  $\text{type}(\hat{\omega})$  in  $H$  moves to  $(\omega, S)$ .

► **Proposition 14.** *For any  $n$ -partite pure simplicial complex  $(X, \pi)$  and a  $k$ -regular labelled graph  $H = (\binom{[n]}{\ell}, E)$ , writing  $\mathbf{Q}_{n \rightarrow \ell}^\downarrow = \text{Down}_{n \rightarrow \ell}(X, \pi, H)$  and  $\mathbf{Q}_{\ell \rightarrow n}^\uparrow = \text{Up}_{\ell \rightarrow n}(X, \pi, H)$  we have,*

$$\left(\pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}\right) \mathbf{Q}_{n \rightarrow \ell}^\downarrow = \pi_\ell \quad \text{and} \quad \pi_\ell \mathbf{Q}_{\ell \rightarrow n}^\uparrow = \pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}.$$

**Proof.** Let  $(\omega, S) \sim \pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}$  be a random sample. Notice that a random neighbor of  $S$  in  $H$  is still distributed uniformly at random as the uniform distribution stationary for the random walk over a  $k$ -regular graph, q.v. Observation 3. Thus, conditional on  $\omega$ , a single step of  $\mathbf{Q}_n^\downarrow$  ends up restricting  $\omega$  to a random set of coordinates  $S$  – this precisely yields the distribution  $\pi_\ell$ , q.v. Equation (3).

The second statement follows since  $\mathbf{Q}_{\ell \rightarrow n}^\uparrow$  is the adjoint operator, q.v. Proposition 1. ◀

The following is easy to verify,

► **Corollary 15.** *For any  $n$ -partite simplicial complex  $(X, \pi)$  and  $k$ -regular labelled graph  $H = (\binom{[n]}{\ell}, E)$ ,*

- $\text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H^2) = \text{Up}_{\ell \rightarrow n}(X, \pi, H) \cdot \text{Down}_{n \rightarrow \ell}(X, \pi, H),$
- $\text{DownUp}_{n \leftrightarrow \ell}(X, \pi, H) = \text{Down}_{n \rightarrow \ell}(X, \pi, H) \cdot \text{Up}_{\ell \rightarrow n}(X, \pi, H).$

We now summarize several useful properties of the expanderized up- and down-walks,

► **Corollary 16.** *Let  $(X, \pi)$  be an  $n$ -partite complex and  $H = (\binom{[n]}{\ell}, E)$  a  $k$ -regular graph. For any  $\ell \leq n$ , writing  $\mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow} = \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H^2)$ ,  $\mathbf{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow \ell}(X, \pi, H)$   $\mathbf{Q}_{n \rightarrow \ell}^\downarrow = \text{Down}_{n \rightarrow \ell}(X, \pi, H)$  and  $\mathbf{Q}_{\ell \rightarrow n}^\uparrow = \text{Up}_{\ell \rightarrow n}(X, \pi, H)$  we have,*

1.  $(\pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}) \mathbf{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}$ , i.e.  $\pi_n \otimes \text{uni}_{\binom{[n]}{\ell}}$  is the stationary distribution of  $\mathbf{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow}$ ,
2.  $\pi_\ell \mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow} = \pi_\ell$ , i.e.  $\pi_\ell$  is the stationary distribution of  $\mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$ .

3.  $Q_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  and  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  are PSD operators.
4.  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  and  $Q_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  are self-adjoint operators.

Since in our proofs it will be more convenient to use  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H)$  directly, initialized with  $H$  and not  $H^2$ , we also note the following.

► **Proposition 17.** <sup>6</sup> Let  $(X, \pi)$  be an  $n$ -partite complex and  $H = (\binom{[n]}{\ell}, E)$  a  $k$ -regular graph. Then, the expanderized up-down walk  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H)$  has the stationary distribution  $\pi_\ell$  and is reversible.

Now, we present the results we prove for expanderized random walks. Our first result shows that the expanderized up-down walk approximates the usual up-down walk in the operator norm,

► **Theorem 18.** Let  $(X, \pi)$  be an  $n$ -partite simplicial complex and let  $H$  be a  $k$ -regular labelled graph on the vertex set  $\binom{[n]}{\ell}$ . Writing  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}(X, \pi, H)$  and  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}(X, \pi)$  for the expanderized- and the regular up-down walks on  $X^{(\ell)}$ , we have

$$\left\| Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} - (1 - \lambda(H)) \cdot P_{\ell \leftrightarrow n}^{\uparrow \downarrow} \right\|_{\text{op}, \pi_\ell} \leq \lambda(H).$$

We present the proof of Theorem 18 in Section 3.1. Theorem 18 immediately implies the following bounds for the spectral gap of expanderized walks,

► **Corollary 19.** <sup>7</sup> Let  $(X, \pi)$  be an  $n$ -partite simplicial complex and let  $H$  be a  $k$ -regular labelled graph on the vertex set  $\binom{[n]}{\ell}$ . Writing  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}(X, \pi, H)$ ,  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} = \text{UpDown}_{\ell \leftrightarrow n}(X, \pi)$ ,  $Q_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \text{DownUp}(X, \pi, H)$ , and  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow \ell}(X, \pi)$ , we have

$$\begin{aligned} \text{gap}\left(Q_{\ell \leftrightarrow n}^{\uparrow \downarrow}\right) &\geq \text{gap}\left(P_{\ell \leftrightarrow n}^{\uparrow \downarrow}\right) \cdot \text{gap}^*(H), \\ \text{gap}\left(Q_{n \leftrightarrow \ell}^{\downarrow \uparrow}\right) &\geq \text{gap}\left(P_{n \leftrightarrow \ell}^{\downarrow \uparrow}\right) \cdot \text{gap}^*(H^2), \end{aligned}$$

where  $\text{gap}^*(G) = 1 - \lambda(G)$  and  $\lambda(G)$  denotes the two-sided expansion of the graph  $G$ .

Unfortunately, a bound on the spectral gap is in many settings not enough to obtain optimal mixing time bounds. We show however, that Theorem 18 allows us to transfer log-Sobolev inequalities (LSI) for the usual up-down walks to the expanderized up-down walks,

► **Corollary 20.** Let  $(X, \pi)$  be an  $n$ -partite simplicial complex and let  $H$  be a  $k$ -regular labelled graph on the vertex set  $\binom{[n]}{\ell}$ . Writing  $Q_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H)$  and  $P_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi)$  for the up-down walk on  $X^{(\ell)}$ , we have

$$\text{ls}\left(Q_{\ell \leftrightarrow n}^{\uparrow \downarrow}\right) \geq \text{ls}\left(P_{\ell \leftrightarrow n}^{\uparrow \downarrow}\right) \cdot \text{gap}^*(H),$$

where  $\text{gap}^*(H) = 1 - \lambda(H)$  and  $\lambda(H)$  denotes the two-sided expansion of the graph  $H$ .

We will prove Corollary 20 in Section 3.2. We state a convenient corollary of Corollary 20 which immediately follows from Lemma 7, Corollary 16, and the data processing inequality Lemma 6,

<sup>6</sup> As Corollary 16 reaches the same end by replacing  $H$  with  $H^2$  we will omit this proof and refer the reader to the full version of our paper [4]

<sup>7</sup> Since this result does not get used in our applications, we refer the reader to the full-version of our paper [4] for a proof.

► **Corollary 21.** *Let  $(X, \pi)$  be an  $n$ -partite simplicial complex and let  $H$  be a  $k$ -regular labelled graph on the vertex set  $\binom{[n]}{\ell}$ . Then, writing  $\mathbf{Q}_{n \leftrightarrow \ell}^\downarrow = \text{Down}_{n \rightarrow \ell}(X, \pi, H)$  and  $\mathbf{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow} = \text{UpDown}_{\ell \leftrightarrow n}(X, \pi)$  we have,*

$$\text{ec}(\mathbf{Q}_{n \leftrightarrow \ell}^\downarrow) \geq \text{ls}(\mathbf{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}) \cdot \text{gap}^*(H^2).$$

*In particular, we have for  $\mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow} := \text{UpDown}_{\ell \leftrightarrow n}(X, \pi, H^2)$  and  $\mathbf{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow \ell}(X, \pi, H)$ ,*

$$\text{ec}(\mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow}) \geq \text{ls}(\mathbf{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}) \cdot \text{gap}^*(H^2) \quad \text{and} \quad \text{ec}(\mathbf{Q}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \geq \text{ls}(\mathbf{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}) \cdot \text{gap}^*(H^2),$$

*where  $\text{gap}^*(H^2) = 1 - \lambda(H^2)$  and  $\lambda(H^2)$  denotes the two-sided expansion of the graph  $H^2$ .*

As we will see in Section 5, Corollary 21 will indeed allow us to prove optimal mixing time bounds for the expanderized walks in many cases of interest.

### 3.1 Closeness in Operator Norm: Proof of Theorem 18

**Proof of Theorem 18 .** For convenience, we will write  $\mathbf{Q}^{\uparrow \downarrow} := \mathbf{Q}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$  and  $\mathbf{P}^{\uparrow \downarrow} := \mathbf{P}_{\ell \leftrightarrow n}^{\uparrow \downarrow}$ . Let  $\mathbf{M}$  denote the random-walk matrix of the graph  $H$  where each transition occurs with the probability  $1/k$  and  $\mathbf{J}$  the random-walk matrix of the clique over  $\binom{[n]}{\ell}$  with self-loops, i.e.  $\mathbf{J} = \mathbf{1}\mathbf{1}^\top / \binom{[n]}{\ell}$ . We will write,  $\lambda := \lambda(\mathbf{M})$ .

Let  $S \in \binom{[n]}{\ell}$  be arbitrary and suppose some  $\bar{\omega} \in X^{(\ell)}$  is given such that  $\text{type}(\bar{\omega}) = S$ .

For all  $\mathbf{f} \in \mathbb{R}^{X^{(\ell)}}$ , we have

$$[\mathbf{Q}^{\uparrow \downarrow} \mathbf{f}](\bar{\omega}) = \sum_{\hat{\omega} \in X[S^c]} \Pr_{\omega \sim \pi}[\omega_{S^c} = \hat{\omega} \mid \omega_S = \bar{\omega}] \cdot \sum_{a \in [k]} \frac{\mathbf{f}((\bar{\omega} \oplus \hat{\omega})_{\text{out}_H(S,a)})}{k}.$$

Similarly, we have

$$[\mathbf{P}^{\uparrow \downarrow} \mathbf{f}](\bar{\omega}) = \sum_{\hat{\omega} \in X[S^c]} \Pr_{\omega \sim \pi}[\omega_{S^c} = \hat{\omega} \mid \omega_S = \bar{\omega}] \sum_{T \in \binom{[n]}{\ell}} \frac{\mathbf{f}((\bar{\omega} \oplus \hat{\omega})_T)}{\binom{[n]}{\ell}}.$$

For any given facet  $\omega \in X^{(n)}$  we define the function  $\mathbf{g}_\omega \in \mathbb{R}^{\binom{[n]}{\ell}}$  as,  $\mathbf{g}_\omega(T) = \mathbf{f}(\omega_T)$ .

We have,

$$[\mathbf{M}\mathbf{g}_\omega](T) = \sum_{a \in [k]} \frac{\mathbf{f}(\omega_{\text{out}_H(T,a)})}{k} \quad \text{and} \quad [\mathbf{J}\mathbf{g}_\omega](i) = \sum_{T \in \binom{[n]}{\ell}} \frac{\mathbf{f}(\omega_T)}{\binom{[n]}{\ell}}.$$

Thus, we have

$$[\mathbf{Q}^{\uparrow \downarrow} \mathbf{f}](\bar{\omega}) = \sum_{\hat{\omega} \in X[S^c]} \Pr_{\omega \sim \pi}[\omega_{S^c} = \hat{\omega} \mid \omega_S = \bar{\omega}] \cdot [\mathbf{M}\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](S), \quad (5)$$

$$[\mathbf{P}^{\uparrow \downarrow} \mathbf{f}](\bar{\omega}) = \sum_{\hat{\omega} \in X[S^c]} \Pr_{\omega \sim \pi}[\omega_{S^c} = \hat{\omega} \mid \omega_S = \bar{\omega}] \cdot [\mathbf{J}\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](S). \quad (6)$$

In particular combining Equation (5) and Equation (6) and noticing that for  $\omega \sim \pi$  the law of  $\omega_{S^c}$  conditional on  $\omega_S = \bar{\omega}$  is given by  $\pi_{n-\ell}^{(\bar{\omega})}$ ,

$$\begin{aligned} \|(Q^{\uparrow\downarrow} - (1-\lambda)P^{\uparrow\downarrow})\mathbf{f}\|_{\pi_\ell}^2 &= \mathbb{E}_{\bar{\omega} \sim \pi_\ell} \left( \mathbb{E}_{\hat{\omega} \sim \pi_{n-\ell}^{(\bar{\omega})}} [ [M\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](\text{type}(\bar{\omega})) - (1-\lambda)[J\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](\text{type}(\bar{\omega})) ] \right)^2, \\ &\leq \mathbb{E}_{\bar{\omega} \sim \pi_\ell} \mathbb{E}_{\hat{\omega} \sim \pi_{n-\ell}^{(\bar{\omega})}} \left[ \left( [M\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](\text{type}(\bar{\omega})) - (1-\lambda)[J\mathbf{g}_{\bar{\omega} \oplus \hat{\omega}}](\text{type}(\bar{\omega})) \right)^2 \right]. \end{aligned}$$

where the last inequality is obtained by appealing to Jensen's inequality and the convexity of  $t \rightarrow t^2$ .

Now, we observe the law of  $\bar{\omega} \oplus \hat{\omega}$  obtained by first sampling  $\bar{\omega} \sim \pi_\ell$  and then  $\hat{\omega} \sim \pi_{n-\ell}^{(\bar{\omega})}$  is given by  $\pi$ . Furthermore, any  $\omega \in X^{(n)}$  occurs exactly  $\binom{[n]}{\ell}$  times in the expectation above – once for each  $S \in \binom{[n]}{\ell}$  acting as  $\text{type}(\bar{\omega})$ , which happens with probability  $\binom{[n]}{\ell}^{-1}$ . Thus,

$$\begin{aligned} \|(Q^{\uparrow\downarrow} - (1-\lambda)P^{\uparrow\downarrow})\mathbf{f}\|_{\pi_\ell}^2 &\leq \mathbb{E}_{\omega \sim \pi_n} \left[ \mathbb{E}_{S \sim \text{uni}^{(\binom{[n]}{\ell})}} \left( [(M - (1-\lambda)J)\mathbf{g}_\omega](S) \right)^2 \right], \\ &\leq \mathbb{E}_{\omega \sim \pi_n} \left[ \lambda^2 \cdot \|\mathbf{g}_\omega\|_{\text{uni}^{(\binom{[n]}{\ell})}}^2 \right] \end{aligned}$$

where the last inequality is due to  $\|M - (1-\lambda)J\|_{\text{op,uni}^{(\binom{[n]}{\ell})}} \leq \lambda$  as  $\lambda(M) \leq \lambda$ !

Now, we finally note

$$\mathbb{E}_{\omega \sim \pi} \left[ \|\mathbf{g}_\omega\|_{\text{uni}^{(\binom{[n]}{\ell})}}^2 \right] = \mathbb{E}_{\omega \sim \pi} \left[ \frac{1}{\binom{[n]}{\ell}} \sum_{S \in \binom{[n]}{\ell}} \mathbf{f}(\omega_S)^2 \right] = \mathbb{E}_{\bar{\omega} \sim \pi_\ell} \mathbf{f}(\bar{\omega})^2 = \|\mathbf{f}\|_{\pi_\ell}^2,$$

The last equality is due to the observation that first sampling  $\omega \sim \pi$  and then outputting  $\omega_S$  for  $S \sim \text{uni}^{(\binom{[n]}{\ell})}$  picked uniformly at random amounts to simply sampling  $\bar{\omega} \sim \pi_\ell$ , q.v. Equation (3).

In particular,

$$\|(Q^{\uparrow\downarrow} - (1-\lambda)P^{\uparrow\downarrow})\mathbf{f}\|_{\pi_\ell} \leq \lambda \cdot \|\mathbf{f}\|_{\pi_\ell}.$$

As  $\mathbf{f}$  was picked arbitrarily, this allows us to conclude the proof of our theorem by appealing to the definition of the operator norm.  $\blacktriangleleft$

### 3.2 Log-Sobolev Bound: Proof of Corollary 20

**Proof of Corollary 20.** Let  $\mathbf{f} \in \mathbb{R}_{\geq 0}^{X^{(\ell)}}$  be an arbitrary function satisfying  $\text{Ent}_{\pi_\ell}(\mathbf{f}^2) \neq 0$ . We have,

$$\frac{\langle \mathbf{f}, (I - Q_{\ell \leftrightarrow n}^{\uparrow\downarrow})\mathbf{f} \rangle_{\pi_\ell}}{\text{Ent}_{\pi_\ell}(\mathbf{f})^2} = \frac{\langle \mathbf{f}, (I - (1-\lambda(H)) \cdot P_{\ell \leftrightarrow n}^{\uparrow\downarrow})\mathbf{f} \rangle_{\pi_\ell}}{\text{Ent}_{\pi_\ell}(\mathbf{f}^2)} + \frac{\langle \mathbf{f}, ((1-\lambda(H))P_{\ell \leftrightarrow n}^{\uparrow\downarrow} - Q_{\ell \leftrightarrow n}^{\uparrow\downarrow})\mathbf{f} \rangle_{\pi_\ell}}{\text{Ent}_{\pi_\ell}(\mathbf{f}^2)}.$$

Notice that by Theorem 18, we should have

$$\langle \mathbf{f}, \left( (1-\lambda(H))P_{\ell \leftrightarrow n}^{\uparrow\downarrow} - Q_{\ell \leftrightarrow n}^{\uparrow\downarrow} \right) \mathbf{f} \rangle_{\pi_\ell} \geq -\lambda(H) \cdot \langle \mathbf{f}, \mathbf{f} \rangle_{\pi_\ell}.$$

Thus,

$$\begin{aligned} \frac{\langle \mathbf{f}, (I - Q_{\ell \leftrightarrow n}^{\uparrow\downarrow})\mathbf{f} \rangle_{\pi_\ell}}{\text{Ent}_{\pi_\ell}(\mathbf{f})^2} &\geq \frac{\langle \mathbf{f}, \left( I - \left( (1-\lambda(H)) \cdot P_{\ell \leftrightarrow n}^{\uparrow\downarrow} + \lambda(H) \cdot I \right) \right) \mathbf{f} \rangle_{\pi_\ell}}{\text{Ent}_{\pi_\ell}(\mathbf{f}^2)}, \\ &\geq \mathbf{1s} \left( (1-\lambda(H)) \cdot P_{\ell \leftrightarrow n}^{\uparrow\downarrow} + \lambda(H) \cdot I \right), \\ &= \mathbf{1s}(P_{\ell \leftrightarrow n}^{\uparrow\downarrow}) \cdot \text{gap}^*(H), \end{aligned}$$

where the last inequality is obtained by noticing:

$$\langle \mathbf{f}, (I - (a \cdot I + (1 - a)P))\mathbf{f} \rangle_\mu = (1 - a) \cdot \langle \mathbf{f}, (I - P)\mathbf{f} \rangle_\mu,$$

and the variational formula for the log-Sobolev constant (Equation (2)) . Appealing to the definition of the log-Sobolev inequality (LSI) once again yields the result. ◀

#### 4 Functional Inequalities on Simplicial Complexes

In this section, we will prove several functional inequalities involving the down-up walk  $P_{n \leftrightarrow \ell}^{\downarrow \uparrow}$ . For convenience we define the set  $\mathcal{C}_\ell(X)$  as the set of  $\ell$ -chains in  $X$ , i.e. the collection of sequences

$$\emptyset := \omega^{(0)} \subsetneq \omega^{(1)} \subsetneq \dots \subsetneq \omega^{(\ell)} \in X^{(\ell)},$$

such that  $\omega^{(i)} \in X^{(i)}$  for all  $i = 0, \dots, \ell$ . Similarly, for  $x \in X^{(1)}$  we define  $\mathcal{C}_\ell(x)$  as the set of  $\ell$ -chains in  $X$  starting from  $x \in X^{(1)}$ , i.e. the collection of sequences

$$x =: \omega^{(1)} \subsetneq \omega^{(2)} \subsetneq \dots \subsetneq \dots \subsetneq \omega^{(\ell)},$$

such that  $\omega^{(i)} \in X^{(i)}$  for all  $i = 0, \dots, \ell$ .

► **Theorem 22.** *For all  $n$ -partite simplicial complexes  $(X, \pi)$  and convex  $\Phi : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ , we have:*

$$\mathbf{cf}_\Phi(P_{\ell \rightarrow n}^\uparrow) \geq \min \left\{ \prod_{j=0}^{\ell-1} (1 - \mathbf{1c}_\Phi(\omega^{(j)})) \mid \emptyset =: \omega^{(0)} \subsetneq \omega^{(1)} \subsetneq \dots \subsetneq \omega^{(\ell)} \in \mathcal{C}_\ell(X) \right\}. \quad (7)$$

In particular, writing  $\mathbf{1c}_\Phi^{(i)}(X, \pi) = \max_{\hat{\omega} \in X^{(i)}} \mathbf{1c}_\Phi(\hat{\omega})$ , we have

$$\mathbf{cf}_\Phi(P_{\ell \rightarrow n}^\uparrow) \geq \prod_{j=0}^{\ell-1} (1 - \mathbf{1c}_\Phi^{(j)}(X, \pi)).$$

As mentioned before our proof is inspired by the exposition in [14] and follows the Garland method, [31]. After submitting our results to arxiv, it came to our attention that the same proof technique for proving a weaker version of Theorem 22 already appeared in [42] in the context of variance contraction. We will list a few immediate consequences of Theorem 22. The following bound is immediate given Proposition 10 and Theorem 22,

► **Corollary 23 (Spectral Gap Bound).** *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . We have,*

$$\mathbf{gap}(P_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \geq \frac{n - \ell}{n} \cdot \min \left\{ \prod_{i=0}^{\ell-1} \mathbf{gap}(M_{z_i}) \mid \emptyset =: \omega^{(0)} \subsetneq \omega^{(1)} \subsetneq \dots \subsetneq \omega^{(\ell)} \in \mathcal{C}_{\ell-1}(X) \right\}. \quad (8)$$

In particular, writing  $\mathbf{gap}_k(X, \pi) := \min_{x \in X^{(k)}} \mathbf{gap}(M_x)$  we have

$$\mathbf{gap}(P_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \geq \frac{n - \ell}{n} \cdot \prod_{i=0}^{\ell-1} \mathbf{gap}_i(X, \pi).$$

We also prove a useful lemma that shows we can directly relate the entropy contraction constant to the log-Sobolev constant of the down-up walk,

► **Lemma 24.** *Let  $(X, \pi)$  be a simplicial complex of rank  $n$ . For any  $\hat{\omega} \in X$ , we set*

$$\pi_{\hat{\omega}, k}^* = \min_{\tilde{\omega} \in X_{\hat{\omega}}^{(k)}} \pi_k^*(\tilde{\omega}), \quad \text{gap}_{n-2}(X, \pi) = \min_{\hat{\omega} \in X^{(n-2)}} \text{gap}(\mathbf{M}_{\hat{\omega}}), \quad \text{and}$$

$$C_{\hat{\omega}, k} = \begin{cases} 1 & \pi_{\hat{\omega}, k}^* > 1/2, \\ \frac{1-2\pi_{\hat{\omega}, k}^*}{\log\left(\left(\pi_{\hat{\omega}, k}^*\right)^{-1}-1\right)} & \text{otherwise.} \end{cases}$$

where  $\mathbf{M}_{\hat{\omega}}$  is the link of  $\hat{\omega}$  and  $\text{gap}(\bullet)$  denotes the spectral gap. We have,

$$\mathbf{1s}(\mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \geq \min\{C_{\omega^{(\ell)}, n-\ell} \mid \omega^{(\ell)} \in X^{(\ell)}\} \cdot \text{ec}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow}),$$

$$\mathbf{1s}(\mathbf{P}_{n-1}^{\downarrow \uparrow}) \geq \frac{n-1}{n} \cdot \min\{C_{\omega^{(n-2)}, 1} \mid \omega^{(n-2)} \in X^{(n-2)}\} \cdot \text{gap}_{n-2}(X, \pi) \cdot \text{ec}(\mathbf{P}_{n-2 \rightarrow n-1}^{\uparrow}).$$

In particular, writing  $\mathbf{1ec}_i(X, \pi) := \min_{\hat{\omega} \in X^{(i)}} \mathbf{1ec}(\hat{\omega})$  and  $C_{\ell, k} = \min_{\hat{\omega} \in X^{(\ell)}} C_{\hat{\omega}, k}$ ,

$$\mathbf{1s}(\mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \geq C_{\ell, n-\ell} \cdot \prod_{i=0}^{\ell-1} (1 - \mathbf{1ec}_i(X, \pi))$$

We will prove this result in Section 4.2.

#### 4.1 Proof of $\Phi$ -Entropy Contraction Bounds, Theorem 22

**Proof of Theorem 22.** For  $\ell = 0$ , the LHS is equal to 0, thus we see the product in Equation (7) is taken over an empty set and equals 1. Thus, equality holds in this case with  $\text{cf}_{\Phi}(\mathbf{P}_{0 \rightarrow n}^{\uparrow}) = 1$ . We proceed by induction on the rank of the simplicial complex. We have by the chain rule for  $\Phi$ -entropy (Lemma 11),

$$\text{Ent}_{\pi_{\ell}}^{\Phi}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}) = \mathbb{E}_{x \sim \pi_1} \text{Ent}_{\pi_{\ell-1}^{(x)}}^{\Phi}(\mathbf{P}_{x, \ell-1 \rightarrow n-1}^{\uparrow} \mathbf{f}|x) + \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}),$$

where we have used (i)  $(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f})_x(\omega) = [\mathbf{P}_{x, \ell-1 \rightarrow n-1}^{\uparrow} \mathbf{f}|x](\omega \setminus x)$  and (ii) that  $\mathbf{P}_{\ell \rightarrow n}^{\uparrow}$  is row-stochastic, i.e.  $\mathbb{E}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}) = \mathbb{E} \mathbf{f}$ . Let  $c := \min_{x \sim X^{(1)}} \text{cf}_{\Phi}(\mathbf{P}_{x, \ell-1 \rightarrow n-1}^{\uparrow})$ . By the induction hypothesis,

$$c \geq \min \left\{ \prod_{j=1}^{\ell-1} (1 - \mathbf{1c}_{\Phi}(\omega^{(j)})) \mid x \in X^{(1)}, x =: \omega^{(1)} \subsetneq \omega^{(2)} \subsetneq \dots \subsetneq \omega^{(\ell-1)} \in \mathcal{C}_{\ell-1}(x) \right\}. \quad (9)$$

Hence, we obtain,

$$\text{Ent}_{\pi_{\ell}}^{\Phi}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}) \leq (1-c) \mathbb{E}_{x \sim \pi_1} \text{Ent}_{\pi_{n-1}^{(x)}}^{\Phi}(\mathbf{f}|x) + \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}).$$

Now, using the chain-rule (Lemma 11) for  $\Phi$ -entropy once more, we have  $\mathbb{E}_{x \sim \pi_1} \text{Ent}_{\pi_{n-1}^{(x)}}^{\Phi}(\mathbf{f}|x) = \text{Ent}_{\pi_n}^{\Phi}(\mathbf{f}) - \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f})$ . Substituting this in the inequality above, we obtain:

$$\begin{aligned} \text{Ent}_{\pi_{\ell}}^{\Phi}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}) &\leq (1-c) \cdot \left( \text{Ent}_{\pi_n}^{\Phi}(\mathbf{f}) - \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}) \right) + \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}), \\ &= (1-c) \cdot \text{Ent}_{\pi_n}^{\Phi}(\mathbf{f}) + c \cdot \text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}). \end{aligned}$$

Now, using  $\text{Ent}_{\pi_1}^{\Phi}(\mathbf{P}_{1 \rightarrow n}^{\uparrow} \mathbf{f}) \leq \mathbf{1c}_{\Phi}(\emptyset) \cdot \text{Ent}_{\pi_n}^{\Phi}(\mathbf{f})$  we obtain

$$\text{Ent}_{\pi_{\ell}}^{\Phi}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}) \leq (1-c \cdot (1 - \mathbf{1c}_{\Phi}(\emptyset))) \cdot \text{Ent}_{\pi_n}^{\Phi}(\mathbf{f}).$$

The statement now follows from Equation (9) and the definition of the  $\Phi$ -entropy contraction factor. ◀

## 4.2 Proof of the log-Sobolev Inequality, Lemma 24

**Proof of Lemma 24.** We follow a similar strategy to what we have followed to establish Theorem 22. We have,

$$\begin{aligned}
\langle \mathbf{f}, (1 - \mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \mathbf{f} \rangle_{\pi_n} &= \langle \mathbf{f}, \mathbf{f} \rangle_{\pi_n} - \langle \mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}, \mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f} \rangle_{\pi_\ell}, \\
&= \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \left[ \langle \mathbf{f} | \hat{\omega}, \mathbf{f} | \hat{\omega} \rangle_{\pi_{n-\ell}^{(\hat{\omega})}} - \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \left[ \langle \mathbf{P}_{\hat{\omega}, 0 \rightarrow n-\ell}^{\uparrow} \mathbf{f} | \hat{\omega}, \mathbf{P}_{\hat{\omega}, 0 \rightarrow n-\ell}^{\uparrow} \mathbf{f} | \hat{\omega} \rangle_{\pi_{n-\ell}^{(\hat{\omega})}} \right] \right], \\
&= \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \left[ \langle \mathbf{f} | \hat{\omega}, (1 - \mathbf{P}_{\hat{\omega}, n-\ell \leftrightarrow 0}^{\downarrow \uparrow}) \mathbf{f} | \hat{\omega} \rangle_{\pi_{n-\ell}^{(\hat{\omega})}} \right], \\
&= \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \left[ \left\langle \mathbf{f} | \hat{\omega}, \left( 1 - \mathbf{J}_{\pi_{n-\ell}^{(\hat{\omega})}} \right) \mathbf{f} | \hat{\omega} \right\rangle_{\pi_{n-\ell}^{(\hat{\omega})}} \right],
\end{aligned}$$

where we have used Items (1) and (2) of Lemma 12 to obtain the second equality and have written  $\mathbf{J}_\mu = \mathbf{1}\mu$  for the clique with respect to  $\mu$ .

Now, by Lemma 5, we have  $\mathbf{1s}\left(\mathbf{J}_{\pi_{n-\ell}^{(\hat{\omega})}}\right) \geq C_{\hat{\omega}, n-\ell}$  – where  $C_{\hat{\omega}, n-\ell}$  is defined as in the statement of Lemma 24. Thus, writing  $C_{\ell, n-\ell} := \min_{\hat{\omega} \in X^{(\ell)}} C_{\hat{\omega}, n-\ell}$ , we have

$$\langle \mathbf{f}, (1 - \mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \mathbf{f} \rangle_{\pi_n} \geq C_{\ell, n-\ell} \cdot \mathbb{E}_{\hat{\omega} \sim \pi_\ell} \text{Ent}_{\pi_{n-\ell}^{(\hat{\omega})}}(\mathbf{f}^2 | \hat{\omega}), \geq C_{\ell, n-\ell} \cdot (\text{Ent}_{\pi_n}(\mathbf{f}^2) - \text{Ent}_{\pi_\ell}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}^2))$$

where we have used the chain rule for entropy, Lemma 11, to obtain the last statement.

Now, using the definition of  $\Phi$ -entropy contraction, i.e. writing for  $\Phi(t) = t \cdot \log t$ ,

$$\text{Ent}_{\pi_\ell}(\mathbf{P}_{\ell \rightarrow n}^{\uparrow} \mathbf{f}^2) \leq \left( 1 - \text{ec}\left(\mathbf{P}_{\ell \rightarrow n}^{\uparrow}\right) \right) \cdot \text{Ent}_{\pi_n}(\mathbf{f}^2).$$

Thus,

$$\langle \mathbf{f}, (1 - \mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}) \mathbf{f} \rangle_{\pi_n} \geq C_{\ell, n-\ell} \cdot \text{ec}\left(\mathbf{P}_{\ell \rightarrow n}^{\uparrow}\right) \cdot \text{Ent}_{\pi_n}(\mathbf{f}^2).$$

Now, the first statement follows by appealing to the definition of the log-Sobolev inequality (LSI) and the log-Sobolev constant (Equation (2)). The second statement concerning  $\mathbf{P}_{n \leftrightarrow \ell}^{\downarrow \uparrow}$  now immediately follows from Theorem 22.

To obtain the log-Sobolev inequality for  $\mathbf{P}_{n-1}^{\uparrow \downarrow}$ , we make use of Items (1) and (3) in Lemma 12 and proceed as above. We have,

$$\langle \mathbf{f}, (1 - \mathbf{P}_{n-1}^{\uparrow \downarrow}) \mathbf{f} \rangle_{\pi_{n-1}} = \frac{n-1}{n} \cdot \mathbb{E}_{\hat{\omega} \sim \pi_{n-2}} \left[ \langle \mathbf{f} | \hat{\omega}, (1 - \mathbf{M}_{\hat{\omega}}) \mathbf{f} | \hat{\omega} \rangle_{\pi_1^{(\hat{\omega})}} \right]$$

Now, appealing to Lemma 5, we obtain  $\mathbf{1s}(\mathbf{M}_{\hat{\omega}}) \geq \text{gap}(\mathbf{M}_{\hat{\omega}}) \cdot C_{\hat{\omega}, 1}$  for all  $\hat{\omega} \in X^{(n-2)}$ . Thus,

$$\begin{aligned}
\langle \mathbf{f}, (1 - \mathbf{P}_{n-1}^{\uparrow \downarrow}) \mathbf{f} \rangle_{\pi_{n-1}} &\geq \frac{n-1}{n} \cdot C_{n-2, 1} \cdot \text{gap}_{n-2}(X, \pi) \cdot \mathbb{E}_{\hat{\omega} \sim \pi_{n-2}} \text{Ent}_{\pi_1^{(\hat{\omega})}}(\mathbf{f}^2 | \hat{\omega}), \\
&= \frac{n-1}{n} \cdot C_{n-2, 1} \cdot \text{gap}_{n-2}(X, \pi) \cdot (\text{Ent}_{\pi_{n-1}}(\mathbf{f}^2) - \text{Ent}_{\pi_\ell}(\mathbf{P}_{n-2 \rightarrow n-1}^{\uparrow} \mathbf{f}^2)), \\
&= \frac{n-1}{n} \cdot C_{n-2, 1} \cdot \text{gap}_{n-2}(X, \pi) \cdot \text{ec}\left(\mathbf{P}_{n-2 \rightarrow n-1}^{\uparrow}\right) \cdot \text{Ent}_{\pi_{n-1}}(\mathbf{f}^2), \tag{10}
\end{aligned}$$

where we have appealed to the chain rule for entropy, Lemma 11, to obtain the first equality.  $\blacktriangleleft$

## 5 Application: Sampling Using the Expanderized Walks

In the present section, present our results concerning the rapid mixing of the expanderized walks for sampling (i) list-colorings and (ii) Ising models with bounded interaction matrix. First, we describe the random sampling problems we are interested in mention the state of

the art sampling results we are interested in expanderizing, and state our results and present a proof of the list coloring chain. Due to space considerations, we only present the proof of our list-coloring result here in Section 5.1 and refer the reader to the full version of our paper in [4] for the proof of our mixing time estimate for the Ising Model.

A list coloring instance  $(G, \mathcal{L})$  consists of a graph  $G = (V, E)$  and a collection of colours  $\mathcal{L} = (L(v))_v$  for every vertex. A valid list coloring of  $(G, \mathcal{L})$  is then a set of pairs  $\{(v, c(v))\}_{v \in V}$  satisfying the following two conditions,

1.  $c(v) \in L(v)$  for all vertices  $v \in V$ ,
2.  $c(u) \neq c(v)$  for all edges  $\{u, v\} \in E$ .

We will write  $(X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})})$  for the simplicial complex of proper list coloring of  $(G, \mathcal{L})$  weighted by the uniform distribution  $(G, \mathcal{L})$  on all list colorings, i.e.

$$X^{(G, \mathcal{L})} = \left\{ \alpha \subset \prod_{v \in V} \{v\} \times L(v) \mid \text{there is a proper list coloring } \chi \text{ of } (G, \mathcal{L}) \text{ such that } \alpha \subset \chi \right\}.$$

We will show that the expanderized walks rapidly mix when sampling list colorings of bounded degree graphs. Further, the lower bound in the number of colors matches with the state of the art – see [12, 41, 10].

► **Theorem 25.** *Let  $(G, \mathcal{L})$  be a list-coloring instance where  $G = (V, E)$  is a graph on  $n$  vertices of maximum degree  $\Delta \leq O(1)$  and  $H_n$  be a labelled graph on  $[n]$  of constant two-sided expansion  $\lambda(H_n)$  bounded away from 1. Then, for some absolute constant  $\varepsilon \approx 10^{-5,8}$  and any  $K = O(1)$ , if  $(11/6 + K)\Delta \geq |L(v)| \geq (11/6 - \varepsilon) \cdot \Delta$  for all vertices  $v \in V$ , the mixing time of the expanderized walks  $\mathcal{Q}_{n-1}^{\downarrow \uparrow} = \text{UpDown}_{\ell \leftrightarrow n}(X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})}, H^2)$  and  $\mathcal{Q}_n^{\downarrow \uparrow} = \text{DownUp}_{n \leftrightarrow \ell}(X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})}, H)$  satisfies,*

$$\tau_{\text{mix}}(\mathcal{Q}_{n-1}^{\downarrow \uparrow}, \varepsilon) \leq C_1 \cdot n(\log n + \log \varepsilon^{-1}) \quad \text{and} \quad \tau_{\text{mix}}(\mathcal{Q}_n^{\downarrow \uparrow}, \varepsilon) \leq C_2 \cdot n(\log n + \log \log \varepsilon^{-1}),$$

where  $C_1$  and  $C_2$  are universal constants not depending on  $n$  but on  $\Delta$ .

► **Remark 26.** By [5], we can pick a constant degree graph as the graph  $H_n$  in the statement of Theorem 25. Thus, a single step of the random walk can be implemented using  $O(1)$ -random bits – making the total number of random bits used in the random walk  $O(n \log n)$ . In contrast, the standard down-up walk or the up-down walk requires  $O(\log n)$  random bits to perform a single step, and  $O(n \log^2 n)$  random bits in total.

We recall that the Ising model  $\mu_{\mathbf{J}, \mathbf{h}} : \{+1, -1\}^n \rightarrow \mathbb{R}_{\geq 0}$  with interaction matrix  $\mathbf{J} \in \mathbb{R}^{n \times n}$  and external field  $\mathbf{h} \in \mathbb{R}^n$  from statistical physics is a probability distribution on the hypercube satisfying,

$$\mu_{\mathbf{J}, \mathbf{h}}(\mathbf{x}) = \frac{\exp\left(\frac{1}{2}\langle \mathbf{x}, \mathbf{J}\mathbf{x} \rangle_{\ell_2} + \langle \mathbf{h}, \mathbf{x} \rangle_{\ell_2}\right)}{Z(\mathbf{J}, \mathbf{h})} \quad \text{where} \quad Z(\mathbf{J}, \mathbf{h}) = \sum_{\mathbf{x} \in \{+1, -1\}^n} \exp\left(\frac{1}{2}\langle \mathbf{x}, \mathbf{J}\mathbf{x} \rangle_{\ell_2} + \langle \mathbf{h}, \mathbf{x} \rangle_{\ell_2}\right) \quad (11)$$

We notice that we can identify any  $\mathbf{x} \in \{+1, -1\}^n$  with a value by using the encoding,

$$\mathbf{x}^{\pm} = \{(i, \mathbf{x}(i)) \mid i \in [n]\}.$$

Thus, we define the simplicial complex  $(X^{(\mathbf{J}, \mathbf{h})}, \mu_{\mathbf{J}, \mathbf{h}})$ , where

$$X^{(\mathbf{J}, \mathbf{h})} = \{\alpha \subset [n] \times \{\pm 1\} \mid \text{for each } i \in [n], \alpha \text{ contains at most one element } (i, x)\}.$$

We show that our expanderize walks mix rapidly assuming that the external field  $\mathbf{h} \in \mathbb{R}^n$  is well-behaved, i.e.  $\|\mathbf{h}\|_{\ell_\infty}$  does not grow with  $n$ ,

<sup>8</sup> See [12].



► **Theorem 27.** Let  $(X^{(\mathbf{J}, \mathbf{h})}, \mu_{\mathbf{J}, \mathbf{h}})$  be the simplicial complex defined above corresponding to the Ising model defined by the interaction matrix  $\mathbf{J} \in \mathbb{R}^{n \times n}$  and external field  $\mathbf{h} \in \mathbb{R}^n$  and  $H_n$  a constant degree graph whose two-sided expansion is a constant bounded away from 1. Under the assumption that  $\mathbf{J}$  is PSD and satisfies  $\|\mathbf{J}\|_{\text{op}} \leq 1$ , the following hold,

$$\begin{aligned} \tau_{\text{mix}}(\mathbb{Q}_{n-1}^{\uparrow\downarrow}, \varepsilon) &\leq \frac{O(\|\mathbf{h}\|_{\ell_\infty}) \cdot n}{(1 - \|\mathbf{J}\|_{\text{op}})^2} (\log(n + \|\mathbf{h}\|_{\ell_1}) + \log \varepsilon^{-1}) \quad \text{and} \\ \tau_{\text{mix}}(\mathbb{Q}_n^{\downarrow\uparrow}, \varepsilon) &\leq \frac{O(\|\mathbf{h}\|_{\ell_\infty}) \cdot n}{(1 - \|\mathbf{J}\|_{\text{op}})^2} (\log(n + \|\mathbf{h}\|_{\ell_1}) + \log \varepsilon^{-1}), \end{aligned}$$

where the  $O(\bullet)$  notation hides a universal constant not depending on  $n, \mathbf{J}$ , or  $\mathbf{h}$ . Furthermore, the term  $(1 - \|\mathbf{J}\|_{\text{op}})^2$  in the denominator can be replaced with  $(1 - \|\mathbf{J}\|_{\text{op}})(1 - \theta)$  if the maximum operator norm of any two-by-two principal submatrix of  $\mathbf{J}$  is  $\theta$ .

► **Remark 28.** By [5], we can pick a constant degree graph as the graph  $H_n$  in the statement of Theorem 27. Thus, ignoring numerical difficulties in simulating biased coins, a single step of the random walk can be implemented using  $O(1)$ -random bits – making the total number of random bits used in the random walk  $O(n \log n)$  when  $\|\mathbf{h}\|_{\ell_\infty} = O(1)$ . In contrast, the standard down-up walk or the up-down walk requires  $O(\log n)$  random bits to perform a single step and  $O(n \log^2 n)$  random bits in total.

### 5.1 List Coloring of Bounded Degree Graphs

We make the following observations about the complex associated to proper list colorings,

► **Proposition 29** (Folklore).<sup>9</sup> Let  $(G = (V, E), \mathcal{L})$  be a list-coloring instance. Let  $K_+, K_- \in \mathbb{N}$  be parameters satisfying  $\deg(v) + K_+ \geq |L(v)| \geq \deg(v) + K_-$  for all  $v \in V$ . Then, writing  $(Y, \pi) := (X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})})$  we have,

$$\lambda_2(M_{\hat{\chi}}) \leq \frac{1}{K_-} \quad \text{for all } \hat{\chi} \in Y^{(n-2)},$$

where  $M_{\hat{\chi}}$  is the link of the face  $\hat{\chi}$ .

Similarly, for any  $\hat{\chi} \in Y^{(n-2)}$ , we have  $\min_{(u, c) \in Y_{\hat{\chi}}^{(1)}} \pi_1^{(\hat{\chi})}(u, c) \geq \frac{K_-}{(\Delta + K_+)^2}$  where  $\Delta = \max_{v \in V} \deg(v)$ .

We recall the following result of [41, 10],

► **Theorem 30** (Theorem 1.2, [41]). Let  $(G, \mathcal{L})$  be a list-coloring instance where  $G = (V, E)$  is a graph on  $n$  vertices of maximum degree  $\Delta \leq O(1)$ . Then, for some absolute constant  $\varepsilon \approx 10^{-5}$ ,<sup>10</sup> if  $|L(v)| \geq (11/6 - \varepsilon) \cdot \Delta$  for all vertices  $v \in V$ , then the spectral gap and the log-Sobolev constants (Equation (2)) of the down-up walk  $\mathbb{P}_n^{\downarrow\uparrow} = \text{DownUp}_{n \leftrightarrow n-1}(X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})})$  on the collection of proper list colorings is  $\Omega(n^{-1})$ .

Then, the following corollary immediately follows by Lemma 7 and Proposition 9,

► **Corollary 31.** Let  $(G, \mathcal{L})$  be a list-coloring instance where  $G = (V, E)$  is a graph on  $n$  vertices of maximum degree  $\Delta \leq O(1)$ . Then, for some absolute constant  $\varepsilon \approx 10^{-5}$ , if  $|L(v)| \geq (11/6 - \varepsilon) \cdot \Delta$  for all vertices  $v \in V$ , then the up-operator  $\mathbb{P}_{n-1 \rightarrow n}^{\uparrow} = \text{Up}_{n-1 \rightarrow n}(X^{(G, \mathcal{L})}, \text{uni}^{(G, \mathcal{L})})$  on the collection of proper list colorings of  $(G, \mathcal{L})$  satisfies  $\text{ec}(\mathbb{P}_{n-1 \rightarrow n}^{\uparrow}) \geq \Omega(n^{-1})$ .

<sup>9</sup> See the full version of this paper [4] for a proof.

<sup>10</sup> See [12]

**Proof of Theorem 25.** Notice that by Lemma 13, Corollary 31 implies that  $\text{ec}(P_{n-2 \rightarrow n-1}^\uparrow) \geq \Omega(n^{-1})$  since by Proposition 29 when  $\Delta = O(1)$ ,  $C_{n-2} = \Omega(1)$ , we have  $\text{gap}_{n-2}(X^{(G,\mathcal{L})}, \text{uni}^{(G,\mathcal{L})}) = \Omega(1)$ , by invoking Lemma 24 we obtain that the up-down walk  $P_{n-1}^{\uparrow\downarrow} = \text{UpDown}_{n-1 \leftrightarrow n}(X^{(G,\mathcal{L})}, \text{uni}^{(G,\mathcal{L})})$  satisfies,  $\text{ls}(P_{n-1}^{\uparrow\downarrow}) \geq \Omega(n^{-1})$ . Then, by Corollary 21 and the assumption that the two-sided expansion  $\lambda(H_n)$  is a constant bounded away from 1, we obtain  $\text{ec}(Q_n^{\uparrow\downarrow}), \text{ec}(Q_{n-1}^{\downarrow\uparrow}) \geq \Omega(n^{-1})$ . The result, concerning mixing times follows using Theorem 8 and the observation that the state space for both walks is of size at most  $n \cdot ((K + 11/6) \cdot \Delta)^n$ . ◀

---

## References

- 1 David Aldous and James Fill. Reversible Markov chains and random walks on graphs, 1995.
- 2 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *STOC*, pages 1198–1211, 2020.
- 3 Vedat Levi Alev and Ori Parzanchevski. Sequential sweeps and high dimensional expansion. *CoRR*, abs/2312.02089, 2023. doi:10.48550/arXiv.2312.02089.
- 4 Vedat Levi Alev and Shravas Rao. Expanderizing higher order random walks, 2024. arXiv:2405.08927.
- 5 Noga Alon. Explicit expanders of every degree and size. *Combinatorica*, 41(4):447–463, 2021. doi:10.1007/s00493-020-4429-x.
- 6 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence ii: optimal sampling and concentration via restricted modified log-sobolev inequalities. *arXiv preprint*, 2021. arXiv:2111.03247.
- 7 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence: optimal mixing of down-up random walks. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1418–1430, 2022.
- 8 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Universality of spectral independence with applications to fast mixing in spin glasses. *CoRR*, abs/2307.10466, 2023. doi:10.48550/arXiv.2307.10466.
- 9 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. *CoRR*, abs/2001.00303, 2020. arXiv:2001.00303.
- 10 Antonio Blanca, Pietro Caputo, Zongchen Chen, Daniel Parisi, Daniel Štefankovič, and Eric Vigoda. On mixing of markov chains: Coupling, spectral independence, and entropy factorization. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3670–3692. SIAM, 2022.
- 11 Antonio Blanca, Pietro Caputo, Daniel Parisi, Alistair Sinclair, and Eric Vigoda. Entropy decay in the swendsen–wang dynamics on zd. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1551–1564, 2021.
- 12 Sitan Chen, Michelle Delcourt, Ankur Moitra, Guillem Perarnau, and Luke Postle. Improved bounds for randomly sampling colorings via linear programming. In *SODA*, pages 2216–2234, 2019. doi:10.1137/1.9781611975482.134.
- 13 Yuansi Chen. An almost constant lower bound of the isoperimetric coefficient in the kls conjecture. *Geometric and Functional Analysis*, 31:34–61, 2021.
- 14 Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for markov chains. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–122. IEEE, 2022.
- 15 Zongchen Chen, Andreas Galanis, Daniel Stefankovic, and Eric Vigoda. Rapid mixing for colorings via spectral independence. *CoRR*, abs/2007.08058, 2020. arXiv:2007.08058.
- 16 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Rapid mixing of Glauber dynamics up to uniqueness via contraction. *arXiv preprint*, 2020. arXiv:2004.09083.

- 17 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*, pages 1537–1550, 2021.
- 18 Persi Diaconis and Arun Ram. Analysis of systematic scan metropolis algorithms using iwahori-hecke algebra techniques. *Michigan Mathematical Journal*, 48(1):157–190, 2000.
- 19 Persi Diaconis, Laurent Saloff-Coste, et al. Logarithmic Sobolev inequalities for finite Markov chains. *The Annals of Applied Probability*, 6(3):695–750, 1996.
- 20 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *APPROX/RANDOM*, pages 38:1–38:20, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.38.
- 21 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *FOCS*, pages 974–985, 2017. doi:10.1109/FOCS.2017.94.
- 22 Roland L Dobrushin. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3):458–486, 1970.
- 23 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Systematic scan for sampling colorings. *The Annals of Applied Probability*, 16(1):185–230, 2006.
- 24 Martin Dyer, Leslie Ann Goldberg, and Mark Jerrum. Dobrushin conditions and systematic scan. *Combinatorics, Probability and Computing*, 17(6):761–779, 2008.
- 25 Ronen Eldan. Thin shell implies spectral gap up to polylog via a stochastic localization scheme. *Geometric and Functional Analysis*, 23(2):532–569, 2013.
- 26 Ronen Eldan. Taming correlations through entropy-efficient measure decompositions with applications to mean-field approximation. *Probability Theory and Related Fields*, 176(3):737–755, 2020.
- 27 Ronen Eldan, Frederic Koehler, and Ofer Zeitouni. A spectral condition for spectral gap: fast mixing in high-temperature ising models. *Probability theory and related fields*, 182(3):1035–1051, 2022.
- 28 Ronen Eldan and Omer Shamir. Log concavity and concentration of lipschitz functions on the boolean hypercube. *Journal of functional analysis*, 282(8):109392, 2022.
- 29 Weiming Feng, Heng Guo, Chunyang Wang, Jiaheng Wang, and Yitong Yin. Towards derandomising markov chain monte carlo. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 1963–1990. IEEE, 2023.
- 30 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Rapid mixing from spectral independence beyond the boolean domain. *arXiv preprint*, 2020. arXiv:2007.08091.
- 31 Howard Garland. p-adic curvature and the cohomology of discrete subgroups of p-adic groups. *Annals of Mathematics*, pages 375–423, 1973.
- 32 Heng Guo and Giorgos Mousa. Local-to-global contraction in simplicial complexes, 2021. arXiv:2012.14317.
- 33 Thomas P Hayes. A simple condition implying rapid mixing of single-site dynamics on spin systems. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 39–46. IEEE, 2006.
- 34 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
- 35 Eyal Karni and Tali Kaufman. High dimensional expansion using zig-zag product. *arXiv preprint*, 2020. arXiv:2001.08829.
- 36 Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. In *ITCS*, pages 4:1–4:27, 2017. doi:10.4230/LIPIcs.ITCS.2017.4.
- 37 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. In *APPROX/RANDOM*, pages 47:1–47:17, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.47.
- 38 B Klartag and V Milman. The slicing problem by bourgain. In *Analysis at Large: Dedicated to the Life and Work of Jean Bourgain*, pages 203–231. Springer, 2022.

- 39 Bo'az Klartag. Eldan's stochastic localization and tubular neighborhoods of complex-analytic sets. *The Journal of Geometric Analysis*, 28:2008–2027, 2018.
- 40 Holden Lee. Parallelising glauber dynamics. *arXiv preprint*, 2023. [arXiv:2307.07131](https://arxiv.org/abs/2307.07131).
- 41 Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. *arXiv preprint*, 2021. [arXiv:2103.11609](https://arxiv.org/abs/2103.11609).
- 42 Kuikui Liu. *Spectral Independence a New Tool to Analyze Markov Chains*. PhD thesis, University of Washington, 2023.
- 43 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Explicit expanders and the ramanujan conjectures. In *STOC*, pages 240–246, 1986. doi:10.1145/12130.12154.
- 44 G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problemy Peredachi Informatsii*, 24(1):51–60, 1988.
- 45 Laurent Miclo. *Remarques sur l'hypercontractivité et l'évolution de l'entropie pour des chaînes de Markov finies*. Springer, 1997.
- 46 Ravi Montenegro and Prasad Tetali. Mathematical aspects of mixing times in Markov chains. *Foundations and Trends in Theoretical Computer Science*, 1(3), 2005. doi:10.1561/0400000003.
- 47 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Derandomization beyond connectivity: Undirected laplacian systems in nearly logarithmic space. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 801–812. IEEE, 2017.
- 48 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic approximation of random walks in small space. *Theory of Computing*, 17(1), 2021.
- 49 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):1–24, 2008.
- 50 Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *FOCS*, pages 3–13, 2000. doi:10.1109/SFCS.2000.892006.
- 51 Gareth O Roberts and Jeffrey S Rosenthal. Surprising convergence properties of some simple gibbs samplers under various scans. *International Journal of Statistics and Probability*, 5(1):51–60, 2015.
- 52 Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 436–447. Springer, 2005.
- 53 Laurent Saloff-Coste. Lectures on finite Markov chains. In *Lectures on probability theory and statistics*, pages 301–413. Springer, 1997.
- 54 Daniel Stefankovic and Eric Vigoda. Lecture notes on spectral independence and bases of a matroid: Local-to-global and trickle-down from a markov chain perspective, 2023. [arXiv:2307.13826](https://arxiv.org/abs/2307.13826).
- 55 EL Wilmer, David A Levin, and Yuval Peres. Markov chains and mixing times. *American Mathematical Soc., Providence*, 2009.

# Ramsey Properties of Randomly Perturbed Hypergraphs

Elad Aigner-Horev   

School of Computer Science, Ariel University, Israel

Dan Hefetz   

School of Computer Science, Ariel University, Israel

Mathias Schacht  

Fachbereich Mathematik, Universität Hamburg, Germany

---

## Abstract

We study Ramsey properties of randomly perturbed 3-uniform hypergraphs. For  $t \geq 2$ , write  $\tilde{K}_t^{(3)}$  to denote the 3-uniform *expanded* clique hypergraph obtained from the complete graph  $K_t$  by expanding each of the edges of the latter with a new additional vertex. For an even integer  $t \geq 4$ , let  $M$  denote the asymmetric maximal density of the pair  $(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ . We prove that adding a set  $F$  of random hyperedges satisfying  $|F| \gg n^{3-1/M}$  to a given  $n$ -vertex 3-uniform hypergraph  $H$  with non-vanishing edge density asymptotically almost surely results in a perturbed hypergraph enjoying the Ramsey property for  $\tilde{K}_t^{(3)}$  and two colours. We conjecture that this result is asymptotically best possible with respect to the size of  $F$  whenever  $t \geq 6$  is even. The key tools of our proof are a new variant of the hypergraph regularity lemma accompanied with a *tuple lemma* providing appropriate control over joint link graphs. Our variant combines the so called strong and the weak hypergraph regularity lemmata.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Random graphs; Mathematics of computing  $\rightarrow$  Hypergraphs

**Keywords and phrases** Ramsey Theory, Smoothed Analysis, Random Hypergraphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.59

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2311.01750> [7]

## 1 Introduction

### 1.1 Ramsey properties of random hypergraphs

Given a distribution  $\mathcal{R}$  over  $n$ -vertex hypergraphs, as well as an  $n$ -vertex hypergraph  $H$ , referred to as the *seed* hypergraph, unions of the form  $H \cup R$  with  $R \sim \mathcal{R}$  define a distribution over the super-hypergraphs of  $H$ , denoted by  $H \cup \mathcal{R}$ . The hypergraphs  $H \cup \mathcal{R}$  are referred to as *random perturbations* of  $H$ . The study of the properties of such hypergraph distributions has its origins in the seminal work of Spielman and Teng [52, 53] who coined the term *Smoothed Analysis* whilst investigating the performance of algorithms on randomly perturbed inputs.

Recently, the paradigm of Smoothed Analysis, originating from Theoretical Computer Science, has captured the attention of numerous researchers in Combinatorics. In the latter avenue, two dominant strands of results have emerged. One strand pertains to the study of the thresholds for the emergence of various spanning and nearly-spanning configurations within such structures (see, e.g., [3, 4, 5, 6, 10, 11, 12, 13, 21, 28, 34, 35, 41]). The second strand pertains to the extremal and Ramsey-type properties (see, e.g., [1, 2, 6, 8, 18, 19, 20, 36, 46]) of such hypergraphs. Our result lies in the latter vein. We recall the arrow notation



© Elad Aigner-Horev, Dan Hefetz, and Mathias Schacht;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 59; pp. 59:1–59:18



Leibniz International Proceedings in Informatics

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

$G \rightarrow (H_1, H_2)$ , signifying the validity of the asymmetric Ramsey statement that every 2-colouring of the edges of  $G$  yields a monochromatic copy of  $H_1$  in the first colour or a monochromatic copy of  $H_2$  in the second colour. Moreover, in the symmetric case when  $H_1 = H_2 = H$  we simply write  $G \rightarrow (H)$ .

Ramsey properties of randomly perturbed graphs were first investigated by Krivelevich, Sudakov, and Tetali [36]. In that work it was shown that  $n^{-2/(t-1)}$  is the threshold for the asymmetric Ramsey property  $G \cup \mathbb{G}(n, p) \rightarrow (K_3, K_t)$ , whenever  $G$  is an  $n$ -vertex graph of edge density  $d \in (0, 1/2)$  independent of  $n$ . The general problem, put forth by Krivelevich et al., of determining the threshold for the property  $G \cup \mathbb{G}(n, p) \rightarrow (K_s, K_t)$ , whenever  $G$  is dense and  $s, t \geq 4$ , was recently (essentially) resolved by Das and Treglown [20]. Those authors showed that  $n^{-1/m_2(K_t, K_{\lceil s/2 \rceil})}$  is the threshold for the property  $G \cup \mathbb{G}(n, p) \rightarrow (K_s, K_t)$ , when  $G$  is a dense  $n$ -vertex graph and  $t \geq s \geq 5$ , where  $m_2(H_1, H_2)$  denotes the asymmetric maximal 2-density of two graphs  $H_1$  and  $H_2$  (see equation (2) for the definition). For other values of  $t$  and  $s$  we also refer to the work of Das and Treglown [20, Theorem 1.7 and Theorem 5(ii)] and for the special case  $s = t = 4$  in addition to the work of Powierski [46, Theorem 1.8].

The aforementioned Ramsey-type results for randomly perturbed dense graphs are formulated for 2-colourings only. This restriction is well-justified. Indeed, suppose that more than two colours are available. The colouring in which the seed is coloured using one colour and the random perturbation is coloured using all the remaining colours, reduces the problem to that of studying the Ramsey property at hand for truly random hypergraphs.

The earlier results [20, 36, 46], as well as our result, stated in Theorem 1 below, are affected by and closely related to research on Ramsey properties in random graphs and hypergraphs (see, e.g., [17, 26, 27, 29, 30, 31, 38, 39, 40, 42, 44, 45, 47, 48, 49, 50]). For random graphs, the thresholds for *symmetric* Ramsey properties are well-understood due to work of Rödl and Ruciński [47, 49]. Minor exceptions for  $F$  being a star forest aside, this work asserts that  $n^{-1/m_2(F)}$  is the threshold for the property  $\mathbb{G}(n, p) \rightarrow (F)$ , where  $m_2(F)$  denotes the maximal 2-density of the given graph  $F$  (see equation (1) below). The 1-statement of the threshold was extended to random  $k$ -uniform hypergraphs by Conlon and Gowers [17] and by Friedgut, Rödl, and Schacht [26]. However, a complete characterisation of the exceptional cases is not yet available and for the progress towards the 0-statement we refer to the work of Nenadov et al. [44] and Gugelmann et al. [27].

The thresholds of asymmetric Ramsey properties in random graphs are the subject of the *Kohayakawa–Kreuter conjecture* [30]. The 1-statement stipulated by this conjecture has been fairly recently verified by Mousset, Nenadov, and Samotij [42] and progress has been made with respect to the corresponding 0-statement by several researchers [27, 29, 38, 40]. Following some progress [14, 37, 42], the conjecture was finally fully resolved by Christoph, Martinsson, Steiner, and Wigderson [15].

## 1.2 Main result

We study Ramsey properties of randomly perturbed hypergraphs; stating our results requires preparation. A hypergraph  $H$  is said to be *linear* if  $|e \cap f| \leq 1$  holds whenever  $e, f \in E(H)$  are distinct. Amongst the linear hypergraphs, *expanded cliques* are of special interest. Given  $t \geq 2$  and  $k \geq 2$ , the  *$k$ -uniformly expanded clique of order  $t$* , denoted by  $\tilde{K}_t^{(k)}$ , is the  $k$ -uniform hypergraph with vertex set of size  $t + \binom{t}{2}(k-2)$  obtained from the complete graph  $K_t$  by expanding every edge of  $K_t$  by  $k-2$  new vertices; in particular,  $\tilde{K}_t^{(2)} = K_t$  holds. Expanded cliques have attracted some attention in the literature and related extremal and Ramsey-type questions were addressed by Mubayi [43] and by Conlon, Fox, and Rödl [16].

Two natural measures of density, arising in the context of random hypergraphs, are the *maximum density* of a  $k$ -uniform  $H = (V, E)$ , denoted  $m(H)$ , and its *maximum  $k$ -density*, denoted  $m_k(H)$ . The former is given by

$$m(H) = \max \left\{ \frac{e(F)}{v(F)} : F \subseteq H \text{ and } v(F) \geq 1 \right\}$$

and the latter is defined by

$$m_k(H) = \max \{d_k(F) : F \subseteq H\}, \text{ where } d_k(F) = \begin{cases} 0, & \text{if } e(F) = 0, \\ \frac{1}{k}, & \text{if } e(F) = 1, v(F) = k, \\ \frac{e(H)-1}{v(H)-k}, & \text{otherwise.} \end{cases} \quad (1)$$

It is well known that  $n^{-1/m(H)}$  is the threshold for the appearance of  $H$  as a subhypergraph in the binomial random  $k$ -uniform hypergraph  $\mathbb{H}^{(k)}(n, p)$ . For  $\mathbb{H}^{(k)}(n, p)$  to satisfy the Ramsey property for  $H$  a.a.s. it is reasonable to expect that many intermingled copies of  $H$  are required; this as to create colour restrictions forcing the Ramsey property for  $H$ . Indeed, for (hypergraph) cliques it is necessary that many cliques sharing a single hyperedge would appear a.a.s. in  $\mathbb{H}^{(k)}(n, p)$ . This results in the higher threshold  $n^{-1/m_k(H)}$  being encountered for Ramsey properties.

For asymmetric Ramsey properties, another notion of hypergraph density arises. This notion traces back to the work of Kohayakawa and Kreuter [30]. Given two  $k$ -uniform hypergraphs  $H_1$  and  $H_2$ , each with at least one edge and satisfying  $m_k(H_1) \geq m_k(H_2)$ , the *asymmetric maximal  $k$ -density* of  $H_1$  and  $H_2$  is given by

$$m_k(H_1, H_2) = m_k(H_2, H_1) = \max \left\{ \frac{e(F)}{v(F) - k + 1/m_k(H_2)} : F \subseteq H_1 \text{ and } e(F) \geq 1 \right\}, \quad (2)$$

where here we do not mean that  $m_k(\cdot, \cdot)$  is symmetric only that in our notation we do not keep track over the location in which  $H_1$ , the hypergraph with the potentially higher  $m_k(\cdot)$ -density is higher, is placed. The equality  $m_k(H, H) = m_k(H)$  is easy to verify.

With the above notation in place, our main contribution can be stated; this can be viewed as a hypergraph extension of the aforementioned results of Das and Treglown [20]. Below we always tacitly assume that  $H_n$  and  $\mathbb{H}^{(3)}(n, p)$  share the same vertex set.

► **Theorem 1 (Main result).** *For every  $d > 0$  and every even integer  $t \geq 4$ , there exists a constant  $C > 0$  such that for every sequence of 3-uniform  $n$ -vertex hypergraphs  $(H_n)_{n \in \mathbb{N}}$  with  $e(H_n) \geq dn^3$  for every  $n \in \mathbb{N}$  we have*

$$\lim_{n \rightarrow \infty} \mathbb{P}(H_n \cup \mathbb{H}^{(3)}(n, p) \rightarrow (\tilde{K}_t^{(3)})) = 1,$$

whenever  $p = p(n) \geq Cn^{-\frac{1}{M}}$  for  $M = m_3(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ .

Our proof of Theorem 1 relies on two main technical results, which are related to the regularity method for hypergraphs. We present these results in Section 1.3-1.4 below.

The proof of Theorem 1 presented here can be adapted for  $k$ -uniform hypergraphs and the asymmetric Ramsey properties  $H_n \cup \mathbb{H}^{(k)}(n, p) \rightarrow (\tilde{K}_s^{(k)}, \tilde{K}_t^{(k)})$  with  $t \geq s$ . For the sake of brevity, we restrict ourselves to 3-uniform hypergraphs and the symmetric case for even  $t$ . In particular, from here on, unless stated otherwise, we use the term hypergraph to mean a 3-uniform hypergraph. We conjecture that Theorem 1 uncovers the threshold for the Ramsey property in question.

► **Conjecture 2.** *For every even integer  $t \geq 6$  there exist constants  $d, c > 0$ , and there exists a sequence of 3-uniform  $n$ -vertex hypergraphs  $(H_n)_{n \in \mathbb{N}}$  with  $e(H_n) \geq dn^3$  for every  $n \in \mathbb{N}$  such that*

$$\lim_{n \rightarrow \infty} \mathbb{P}(H_n \cup \mathbb{H}^{(3)}(n, p) \longrightarrow (\tilde{K}_t^{(3)})) = 0,$$

whenever  $p \leq cn^{-1/M}$  for  $M = m_3(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ .

Conjecture 2 may hold for  $t = 4$  as well. However, this value is excluded due to the distinct behaviour seen in the graph case [20, 46]. The proof of Theorem 1 presented here extends for the asymmetric Ramsey property  $H \longrightarrow (\tilde{K}_t^{(3)}, \tilde{K}_s^{(3)})$  for sufficiently large integers  $t \geq s$  and  $M$  replaced by  $m_3(\tilde{K}_t^{(3)}, \tilde{K}_{\lceil s/2 \rceil}^{(3)})$ . It seems plausible that the corresponding generalisation of Conjecture 2 may also hold.

### 1.3 A tuple lemma for link graphs

A key feature of the regularity method of graphs is the control over joint neighbourhoods in the regular environment provided by Szemerédi’s regularity lemma (see, e.g., Lemma 8 below). For the proof of Theorem 1, we establish a similar lemma in the context of the regularity method for hypergraphs.

For a vertex  $v$  in a hypergraph  $H = (V, E)$ , define the *link graph*  $L_H(v)$  of  $v$  to have vertex set  $V \setminus \{v\}$  and edge set comprised of those pairs of vertices which together with  $v$  form a hyperedge in  $H$ , i.e.,  $E(L_H(v)) = \{uv : uvw \in E\}$ . In particular,  $e(L_H(v))$  is the vertex degree of  $v$  in  $H$  and is also denoted by  $\deg_H(v)$ . Given a graph  $G$  with vertex set  $V(G) = V$  we define the *link graph of  $v$  supported on  $G$*  by

$$L_H(v, G) = E(L_H(v)) \cap E(G).$$

Link graphs are a natural hypergraph extension of vertex neighbourhoods in the context of graphs. A tuple lemma for hypergraphs would have to control the sizes of the intersections of link graphs. In that, given a set of vertices  $U \subseteq V$ , we seek to control the sizes of the *joint link graph* and the *joint link graph supported by  $G$*  given by

$$L_H(U) = \bigcap_{u \in U} L_H(u) \quad \text{and} \quad L_H(U, G) = \bigcap_{u \in U} L_H(u, G),$$

respectively. For a random hypergraph  $H = (V, E)$  with edge density  $d$ , one would expect  $|L_H(U)| \sim d^{|U|} \binom{|V|}{2}$  to hold with high probability. Our tuple lemma asserts that in the regular environment for hypergraphs this random intuition can be transferred to the deterministic situation. (We defer the definitions concerning regular hypergraphs to Section 2.)

► **Proposition 3 (Tuple lemma for joint links).** *For every  $t \geq 2$  and  $\varepsilon, d_3 > 0$ , there exists a  $\delta_3 > 0$  such that for every  $d_2 > 0$  there exist  $\delta_2 > 0$  and  $r \geq 1$  such that the following holds.*

*Let  $H = (X \cup Y \cup Z, E_H)$  be a tripartite hypergraph which is  $(\delta_3, d_3, r)$ -regular with respect to a  $(\delta_2, d_2)$ -triad  $P = (X \cup Y \cup Z, E_P)$ . Then, all but at most  $2\varepsilon|X|^t$  of the  $t$ -tuples of vertices  $X' = \{x_1, \dots, x_t\} \subseteq X$  satisfy*

$$|L_H(X', P) - d_3^t d_2^{2t+1} |Y||Z|| \leq \varepsilon d_2^{2t+1} |Y||Z|. \tag{3}$$

Due to space limitations, our proof of Proposition 3 is omitted and can be found in [7] - the full version of this extended abstract - the former extends to all hypergraph uniformities. Alternatives to Proposition 3 exerting some control over the sizes of joint link graphs of vertex tuples whilst relying on weaker versions of the hypergraph regularity do exist. Such alternatives are established in the extended account [7].



## 1.4 A variant of the hypergraph regularity lemma

The second main technical lemma is a new variant of the hypergraph regularity lemma established in [51]. The necessary definitions are deferred to Section 2.

► **Proposition 4** (Variant of the regularity lemma for hypergraphs). *For every  $\delta_3 > 0$  and functions  $\delta_2: \mathbb{N} \rightarrow (0, 1]$ ,  $r: \mathbb{N}^2 \rightarrow \mathbb{N}$ , and constants  $\ell_0$ ,  $t_0$ , and  $s \in \mathbb{N}$ , there exist  $n_0$  and  $T \in \mathbb{N}$  such that for every  $n \geq n_0$  and every family  $(H_1, \dots, H_s)$  of  $n$ -vertex hypergraphs satisfying  $V = V(H_1) = \dots = V(H_s)$ , there are integers  $t$  and  $\ell$  satisfying  $t \geq t_0$  and  $\ell \geq \ell_0$ , a vertex partition  $\mathcal{V}$  with  $V_1 \cup \dots \cup V_t = V$  and an  $\ell$ -equitable partition  $\mathcal{B}$  with respect to  $\mathcal{V}$  such that the following properties hold.*

(R.1)  $|V_1| \leq |V_2| \leq \dots \leq |V_t| \leq |V_1| + 1$ ,

(R.2) for all  $1 \leq i < j \leq t$  and  $\alpha \in [\ell]$ , the bipartite 2-graph  $B_\alpha^{ij}$  is  $(\delta_2(\ell), 1/\ell)$ -regular,

(R.3)  $H_i$  is  $\delta_2(\ell)$ -weakly regular with respect to  $\mathcal{V}$  for every  $i \in [s]$ , and

(R.4)  $H_i$  is  $(\delta_3, r(t, \ell))$ -regular with respect to  $\mathcal{B}$  for every  $i \in [s]$ .

Due to space limitations, our proof of Proposition 4 is omitted and can be found in [7] - the full version of this extended abstract. In Proposition 4 there is a combination of the environment of the hypergraph regularity lemma [51] (see Lemma 10) and the so-called weak hypergraph regularity lemma (see Lemma 9 below), which is the straightforward extension of Szemerédi's regularity for graphs. A lemma of similar spirit can be found in the work of Allen, Parczyk, and Pfenninger [9].

In the sequel, these hypergraph regularity lemmata are distinguished by referring to these as the *Strong Lemma* and *Weak Lemma*, respectively. The difference between the Strong Lemma and Proposition 4 is Property (R.3). The former, when applied to dense hypergraphs, provides access to triads  $P$  set over a vertex set, say,  $X \cup Y \cup Z$  with respect to which the regularised hypergraphs is  $(\delta_3, d, r)$ -regular. This, in turn, provides  $\zeta$ -weak regularity control for  $\zeta = \delta_3^{1/3}$ , by which we mean the ability to control the hyperedge distribution of the hypergraphs along sets  $X' \subseteq X$ ,  $Y' \subseteq Y$ , and  $Z' \subseteq Z$  satisfying  $|X'| \geq \zeta|X|$ ,  $|Y'| \geq \zeta|Y|$ , and  $|Z'| \geq \zeta|Z|$ .

The added Property (R.3), however, provides weak regularity control over vertex sets with much smaller density. In fact, there the control  $\delta_2$  is allowed to be a function of  $\ell$  and the quantification of the Strong Lemma leads to  $\delta_3 \gg \ell^{-1}$ .

## Organisation

Theorem 1 is proved in Section 3. Various required preliminaries are collected in Section 2. As mentioned above, the proofs of Propositions 3 and 4 are omitted from this account due to space limitations and can be seen in [7] - the full version of this account.

## Notational remark

Throughout, we often write the enumeration of a result in the subscripts of the constants that it presides over. For instance, the constant  $t_0$  in Proposition 4 becomes  $t_4$  and the constant  $\delta_3$  in the same lemma is written  $\delta_4^{(3)}$  and so on. This aids in keeping track of the various constants encountered throughout the proofs.

## 2 Preliminaries

Let  $V$  be a finite set. A partition  $\mathcal{U}$  of  $V$  given by  $V = U_1 \cup \dots \cup U_r$  is said to be *equitable* if  $|U_1| \leq |U_2| \leq \dots \leq |U_r| \leq |U_1| + 1$ . Given an additional partition of  $V$ , namely  $\mathcal{V}$ , of the form  $V = V_1 \cup \dots \cup V_\ell$ , we say that  $\mathcal{V}$  *refines*  $\mathcal{U}$ , and write  $\mathcal{V} \prec \mathcal{U}$ , if for every  $i \in [\ell]$  there

exists some  $j \in [r]$  such that  $V_i \subseteq U_j$  holds. For  $k \geq 2$ , write  $K^{(k)}(\mathcal{U})$  to denote the complete  $|\mathcal{U}|$ -partite  $k$ -uniform hypergraph whose vertex set is  $V$  and whose edge set is given by all sets of  $V^{(k)} = \{K \subseteq V: |K| = k\}$  meeting every member of  $\mathcal{U}$  (termed *cluster* hereafter) in at most one vertex. If  $\mathcal{U} = \{U, U'\}$  consists of only two clusters, then we abbreviate  $K^{(2)}(\mathcal{U})$  to  $K^{(2)}(U, U')$ . We write  $K^{(2)}(V)$  to denote the complete graph whose vertex set is  $V$ .

## 2.1 Graph regularity

Let  $d, \delta > 0$  be given. A bipartite 2-graph  $G = (X \cup Y, E)$  is said to be  $(\delta, d)$ -regular if

$$e_G(X', Y') = d|X'||Y'| \pm \delta|X||Y|$$

holds<sup>1</sup> for every  $X' \subseteq X$  and  $Y' \subseteq Y$ . If  $d$  coincides with the edge density of  $G$ , i.e.  $d = \frac{e(G)}{|X||Y|}$ , then we abbreviate  $(\delta, d)$ -regular to  $\delta$ -regular. It follows directly from the definition that  $G$  is a  $(\delta, d)$ -regular bipartite graph if, and only if, its (bipartite) complement is  $(\delta, 1 - d)$ -regular.

A tripartite 2-graph  $P$  with vertex set  $V(P) = X \cup Y \cup Z$  is said to be a  $(\delta, d)$ -*triad*, if  $P[X, Y]$ ,  $P[Y, Z]$ , and  $P[X, Z]$  are all  $(\delta, d)$ -regular. For a 2-graph  $G$ , let  $\mathcal{K}_3(G)$  denote the family of members of  $V(G)^{(3)}$  spanning a triangle in  $G$ . We shall employ the well known triangle counting lemma (see, e.g., [25, Fact A]).

► **Lemma 5** (Triangle counting lemma). *Let  $d > 0$ , let  $0 < \delta < d/2$ , and let  $P$  be a  $(\delta, d)$ -triad with vertex set  $V(P) = X \cup Y \cup Z$ . Then,*

$$(1 - 2\delta)(d - \delta)^3|X||Y||Z| \leq |\mathcal{K}_3(P)| \leq ((d + \delta)^3 + 2\delta)|X||Y||Z|.$$

In particular, if  $d \leq 1/2$ , then

$$|\mathcal{K}_3(P)| = (d^3 \pm 4\delta)|X||Y||Z| \tag{4}$$

holds. ┘

We shall also use the variant of the triangle counting lemma with only two of the bipartite graphs being regular and its proof is included for completeness.

► **Lemma 6.** *Let  $P = (X \cup Y \cup Z, E_P)$  be a tripartite 2-graph such that  $P[X, Y]$  and  $P[X, Z]$  are both  $(\delta, d)$ -regular. In addition, let  $X' \subseteq X$  be a set of size  $|X'| \geq \delta|X|$ . Then,*

$$(d - \delta)d|X'|e(P[Y, Z]) - 2\delta|X||Y||Z| \leq |\mathcal{K}_3(P, X')| \leq (d + \delta)d|X'|e(P[Y, Z]) + 2\delta|X||Y||Z|$$

holds, where  $\mathcal{K}_3(P, X')$  denotes the set of triangles of  $P$  meeting  $X'$ .

**Proof.** Let  $Y' \subseteq Y$  consist of all vertices  $y \in Y$  satisfying  $\deg_P(y, X') \geq (d - \delta)|X'|$ ; note that  $|Y'| \geq (1 - \delta)|Y|$  holds by Lemma 8. We may then write

$$\begin{aligned} |\mathcal{K}_3(P, X')| &\geq \sum_{y \in Y'} \left( d(d - \delta)|X'| \deg_P(y, Z) - \delta|X||Z| \right) \\ &= d(d - \delta)|X'| \left( \sum_{y \in Y} \deg_P(y, Z) - \sum_{y \in Y \setminus Y'} \deg_P(y, Z) \right) - \sum_{y \in Y'} \delta|X||Z| \\ &\geq d(d - \delta)|X'|e(P[Y, Z]) - d(d - \delta)\delta|X||Y||Z| - \delta|X||Y||Z| \\ &\geq d(d - \delta)|X'|e(P[Y, Z]) - 2\delta|X||Y||Z|. \end{aligned}$$

<sup>1</sup> Given  $x, y, z \in \mathbb{R}$ , we write  $x = y \pm z$  if  $y - z \leq x \leq y + z$ .

Next, we prove the upper bound. Let  $Y'' \subseteq Y$  consist of all vertices  $y \in Y$  satisfying  $\deg_P(y, X') \leq (d + \delta)|X'|$ ; note that  $|Y''| \geq (1 - \delta)|Y|$  holds by Lemma 8. We may then write

$$\begin{aligned} |\mathcal{K}_3(P, X')| &\leq \sum_{y \in Y''} \left( d(d + \delta)|X'| \deg_P(y, Z) + \delta|X||Z| \right) + \sum_{y \in Y \setminus Y''} |X'||Z| \\ &\leq d(d + \delta)|X'| \left( \sum_{y \in Y} \deg_P(y, Z) - \sum_{y \in Y \setminus Y''} \deg_P(y, Z) \right) \\ &\quad + \sum_{y \in Y''} \delta|X||Z| + \sum_{y \in Y \setminus Y''} |X||Z| \\ &\leq d(d + \delta)|X'|e(P[Y, Z]) + 2\delta|X||Y||Z|. \end{aligned} \quad \blacktriangleleft$$

The next lemma is commonly referred to as the *Slicing Lemma* (see, e.g., [33, Fact 1.5]).

► **Lemma 7** (Slicing lemma). *Let  $d = d_7$ , let  $\delta = \delta_7 > 0$ , and let  $G = (A \cup B, E)$  be a  $(\delta, d)$ -regular bipartite graph. Let  $\delta \leq \alpha = \alpha_7 \leq 1$ , and let  $A' \subseteq A$  and  $B' \subseteq B$  be sets of sizes  $|A'| \geq \alpha|A|$  and  $|B'| \geq \alpha|B|$ . Then,  $G[A', B']$  is  $(\delta', d')$ -regular where  $\delta' = \max\{\delta/\alpha, 2\delta\}$  and  $d' = d \pm \delta$ .  $\blacktriangleright$*

The *tuple property* of dense regular bipartite graphs, also referred to as the *intersection property*, reads as follows (see [33, Fact 1.4]).

► **Lemma 8** (Tuple lemma for graphs). *Let  $G = (X \cup Y, E)$  be a  $\delta$ -regular bipartite graph of edge density  $d > 0$ . Then, all but at most  $2\delta\ell|X|^\ell$  of the tuples  $\{x_1, \dots, x_\ell\} \subseteq X$  satisfy*

$$|N_G(x_1, \dots, x_\ell, Y')| = |\{y \in Y' : x_i y \in E(G) \text{ for all } i \in [\ell]\}| = (d \pm \delta)^\ell |Y'|, \quad (5)$$

whenever  $Y' \subseteq Y$  satisfies  $(d - \delta)^{\ell-1}|Y'| \geq \delta|Y|$ .  $\blacktriangleright$

## 2.2 Hypergraph regularity

A direct generalisation of the notion of  $\delta$ -regularity, defined in the previous section for 2-graphs, reads as follows. Let  $d, \delta > 0$ . A tripartite hypergraph  $H = (X \cup Y \cup Z, E)$  is said to be  $(\delta, d)$ -weakly regular if

$$e_H(X', Y', Z') = d|X'||Y'||Z'| \pm \delta|X||Y||Z|$$

holds whenever  $X' \subseteq X$ ,  $Y' \subseteq Y$ , and  $Z' \subseteq Z$ . If  $d = \frac{e(H)}{|X||Y||Z|}$ , then we abbreviate  $(\delta, d)$ -weakly regular to  $\delta$ -weakly regular.

Given a partition  $\mathcal{V}$  of a finite set  $V$  defined by  $V = V_1 \cup \dots \cup V_t$ , a hypergraph  $H$  with  $V(H) = V$  is said to be  $\delta$ -weakly regular with respect to  $\mathcal{V}$  if  $H[X, Y, Z]^2$  is  $\delta$ -weakly regular with respect to all but at most  $\delta \binom{t}{3}$  triples  $\{X, Y, Z\} \in \mathcal{V}^{(3)}$ . We state the straightforward adaptation of Szemerédi's graph regularity lemma [32, 33, 54].

► **Lemma 9** (Weak hypergraph regularity lemma). *For every  $\delta = \delta_9 > 0$  and positive integers  $s = s_9$ ,  $t = t_9$ , and  $h = h_9$  satisfying  $t \geq h$ , there exist positive integers  $n_0$  and  $T = T_9$  such that the following holds whenever  $n \geq n_0$ . Let  $(H_1, \dots, H_s)$  be a sequence of  $n$ -vertex hypergraphs, all on the same vertex set, namely  $V$ , and let  $\mathcal{U} = \mathcal{U}_9$  be a vertex partition of*

<sup>2</sup>  $H[X, Y, Z]$  is the subgraph of  $H$  over  $X \cup Y \cup Z$  whose edge set is  $\{\{x, y, z\} \in E(H) : x \in X, y \in Y, z \in Z\}$ .

$V$  given by  $V = U_1 \cup \dots \cup U_h$ . Then, there exists an equitable vertex partition  $\mathcal{V}$ , given by  $V = V_1 \cup V_2 \cup \dots \cup V_{t'}$ , where  $t \leq t' \leq T$ , such that  $\mathcal{V} \prec \mathcal{U}$  and, moreover,  $H_i$  is  $\delta$ -weakly regular with respect to  $\mathcal{V}$  for every  $i \in [s]$ .  $\lrcorner$

We proceed to the statement of the Strong hypergraph Regularity Lemma for hypergraphs following the formulation seen in [51]. Given a 2-graph  $G$ , the *relative density* of a hypergraph  $H$  with vertex set  $V(H) = V(G)$ , with respect to  $G$  is given by

$$d(H|G) = \frac{|E(H) \cap \mathcal{K}_3(G)|}{|\mathcal{K}_3(G)|}. \quad (6)$$

For  $\delta, d > 0$  and a positive integer  $r$ , a tripartite hypergraph  $H = (X \cup Y \cup Z, E_H)$  is said to be  $(\delta, d, r)$ -regular with respect to a tripartite 2-graph  $P = (X \cup Y \cup Z, E_P)$  if

$$\left| \left| \bigcup_{i=1}^r (E_H \cap \mathcal{K}_3(Q_i)) \right| - d \left| \bigcup_{i=1}^r \mathcal{K}_3(Q_i) \right| \right| \leq \delta |\mathcal{K}_3(P)| \quad (7)$$

holds for every family of, not necessarily disjoint, subgraphs  $Q_1, \dots, Q_r \subseteq P$  satisfying

$$\left| \bigcup_{i=1}^r \mathcal{K}_3(Q_i) \right| \geq \delta |\mathcal{K}_3(P)| > 0.$$

Let  $V$  be a finite set and let  $\mathcal{V}$  be a partition  $V_1 \cup \dots \cup V_h$  of  $V$ , where  $h$  is some positive integer. Given an integer  $\ell \geq 1$ , a partition  $\mathcal{B}$  of  $K^{(2)}(\mathcal{V})$  is said to be  $\ell$ -equitable with respect to  $\mathcal{V}$  if it satisfies the following conditions:

**(B.1)** every  $B \in \mathcal{B}$  satisfies  $B \subseteq K^{(2)}(V_i, V_j)$  for some distinct  $i, j \in [h]$ ; and

**(B.2)** for any distinct  $i, j \in [h]$ , precisely  $\ell$  members of  $\mathcal{B}$  partition  $K^{(2)}(V_i, V_j)$ .

We view partitions of  $K^{(2)}(\mathcal{V})$  as partitions of  $V^{(2)}$  under the *agreement*<sup>3</sup> that the set  $\{K^{(2)}(V_i) : i \in [h]\}$  of complete graphs is added to the former; such an addition of cliques does not hinder the equitability notion defined in (B.2); it does violate (B.1), but this will not harm our arguments. Moreover, it is under this agreement that we say that a partition of  $V^{(2)}$  refines a partition of  $K^{(2)}(\mathcal{V})$ .

For distinct indices  $i, j \in [h]$ , the partition of  $K^{(2)}(V_i, V_j)$  induced by  $\mathcal{B}$  is denoted by

$$\mathcal{B}^{ij} = \{B_\alpha^{ij} = (V_i \cup V_j, E_\alpha^{ij}) : \alpha \in [\ell]\}.$$

The *triads* of  $\mathcal{B}$  are the tripartite 2-graphs having the form

$$B_{\alpha\beta\gamma}^{ijk} = (V_i \cup V_j \cup V_k, E_\alpha^{ij} \cup E_\beta^{ik} \cup E_\gamma^{jk}),$$

where  $i, j, k \in [h]$  are distinct and  $\alpha, \beta, \gamma \in [\ell]$ . Recall that a triad is called a  $(\delta, d)$ -triad if each of the three bipartite graphs comprising it is  $(\delta, d)$ -regular. A hypergraph  $H$  with vertex set  $V(H) = V$  is said to be  $(\delta, r)$ -regular with respect to  $\mathcal{B}$  if

$$\left\{ \bigcup_{\substack{1 \leq i < j < k \leq h \\ \alpha, \beta, \gamma \in [\ell]}} \mathcal{K}_3(B_{\alpha\beta\gamma}^{ijk}) : H_{ijk} \text{ is not } (\delta, d(H|B_{\alpha\beta\gamma}^{ijk}), r)\text{-regular w.r.t. } B_{\alpha\beta\gamma}^{ijk} \right\} \leq \delta |V|^3,$$

where  $H_{ijk} = H[V_i \cup V_j \cup V_k]$ . A formulation of the Strong Lemma [51, Theorem 17] for hypergraphs, reads as follows.

<sup>3</sup> We appeal to this agreement in our proof of Proposition 4 omitted from this account and which can be found in [7].

► **Lemma 10** (Strong hypergraph regularity lemma). *For all  $0 < \delta_3 \in \mathbb{R}$ ,  $\delta_2: \mathbb{N} \rightarrow (0, 1]$ ,  $r: \mathbb{N}^2 \rightarrow \mathbb{N}$ , and  $s, t, \ell \in \mathbb{N}$ , there exist  $n_0, T \in \mathbb{N}$  such that for every  $n \geq n_0$  and every sequence of  $n$ -vertex hypergraphs  $(H_1, \dots, H_s)$ , satisfying  $V = V(H_1) = \dots = V(H_s)$ , there are  $t', \ell' \in \mathbb{N}$  satisfying  $t \leq t' \leq T$  and  $\ell \leq \ell' \leq T$ , a vertex partition  $V = V_1 \cup \dots \cup V_{t'}$ , namely  $\mathcal{V}$ , and an  $\ell'$ -equitable partition  $\mathcal{B}$  with respect to  $\mathcal{V}$  such that the following properties hold.*

- (S.1)  $|V_1| \leq |V_2| \leq \dots \leq |V_{t'}| \leq |V_1| + 1$ ;
- (S.2) for all  $1 \leq i < j \leq t'$  and  $\alpha \in [\ell']$ , the bipartite 2-graph  $B_\alpha^{ij}$  is  $(\delta_2(\ell'), 1/\ell')$ -regular; and
- (S.3)  $H_i$  is  $(\delta_3, r(t', \ell'))$ -regular with respect to  $\mathcal{B}$  for every  $i \in [s]$ . ◻

### 3 Monochromatic expanded cliques

In this section, we prove Theorem 1. The required Ramsey properties of  $\mathbb{H}^{(3)}(n, p)$  are collected in Section 3.1; a proof of Theorem 1 can be found in Section 3.2. For an integer  $t \geq 3$ , the  $t$  vertices of  $\tilde{K}_t^{(k)}$  having their 1-degree strictly larger than one are called the *branch-vertices* of  $\tilde{K}_t^{(k)}$ . Set

$$v(t) = v(\tilde{K}_t^{(3)}) \quad \text{and} \quad e(t) = e(\tilde{K}_t^{(3)}).$$

#### 3.1 Properties of random hypergraphs

The main goal of this section is to state Proposition 11 which is an adaptation of [20, Theorem 2.10]. This proposition collects the Ramsey properties of  $\mathbb{H}^{(3)}(n, p)$  that will be utilised throughout our proof of Theorem 1.

A  $k$ -graph  $H$  is said to be *balanced* if  $m_k(H) = d_k(H)$  holds; if all proper subgraphs  $F$  of  $H$  satisfy  $m_k(F) < m_k(H)$ , then  $H$  is said to be *strictly balanced*. It is not hard to verify that expanded cliques are strictly balanced. In particular,

$$m_k(\tilde{K}_t^{(k)}) = \frac{\binom{t}{2} - 1}{t + (k-2)\binom{t}{2} - k}$$

holds for any  $k \geq 2$  and  $t \geq 3$ . In the special case  $k = 3$  we obtain

$$m_3(\tilde{K}_t^{(3)}) = \frac{t^2 - t - 2}{t^2 + t - 6} = 1 - \frac{2t - 4}{t^2 + t - 6} < 1, \tag{8}$$

that is, 3-uniformly expanded cliques are *sparse*. Note that this is in contrast to graph cliques (on at least 3 vertices) whose 2-density is larger than one. For a simpler notation we set an integer  $t \geq 2$

$$m(t) = m(\tilde{K}_t^{(3)}) \quad \text{and} \quad M_t = m_3(\tilde{K}_t^{(3)}).$$

Similarly for integers  $t_1, t_2 \geq 2$  we set

$$M_{t_1, t_2} = M_{t_2, t_1} = m_3(\tilde{K}_{t_1}^{(3)}, \tilde{K}_{t_2}^{(3)}).$$

Let  $H_1$  and  $H_2$  be two  $k$ -graphs, each with at least one edge and such that  $m_k(H_1) \geq m_k(H_2)$ . If  $m_k(H_1) = m_k(H_2)$ , then  $m_k(H_1, H_2) = m_k(H_1)$ ; otherwise  $m_k(H_2) < m_k(H_1, H_2) < m_k(H_1)$  holds. The  $k$ -graph  $H_1$  is said to be *strictly balanced with respect to  $m_k(\cdot, H_2)$*  if no proper subgraph  $F \subsetneq H_1$  maximises  $\binom{2}{2}$ . For instance, it is not hard to verify that  $\tilde{K}_t^{(3)}$  is strictly balanced with respect to  $m_3(\cdot, \tilde{K}_{t/2}^{(3)})$ , assuming  $t \geq 4$  is even.

Let  $F$  and  $F'$  be  $k$ -graphs and let  $\mu = \mu(n)$  be given. An  $n$ -vertex  $k$ -graph  $H$  is said to be  $(F, \mu)$ -Ramsey if  $H[U] \rightarrow (F)_2$  holds for every  $U \subseteq V(H)$  of size  $|U| \geq \mu n$ . Similarly,  $H$  is said to be  $(F, F', \mu)$ -Ramsey if  $H[U] \rightarrow (F, F')$  holds for every  $U \subseteq V(H)$  of size  $|U| \geq \mu n$ . Given  $\mathcal{F} \subseteq \binom{[n]}{v(F)}$  and  $\mathcal{F}' \subseteq \binom{[n]}{v(F')}$ , we say that  $H$  is  $(F, F')$ -Ramsey with respect to  $(\mathcal{F}, \mathcal{F}')$  if any 2-colouring of  $E(H)$  yields a monochromatic copy  $K$  of  $F$  (in the first colour) with  $V(K) \notin \mathcal{F}$  or a monochromatic copy  $K'$  of  $F'$  (in the second colour) with  $V(K') \notin \mathcal{F}'$ .

► **Proposition 11.** *Let  $t \geq 4$  be an even integer. The binomial random hypergraph  $H \sim \mathbb{H}^{(3)}(n, p)$  a.a.s. satisfies the following properties.*

- (P.1) *There are constants  $\gamma_{11} = \gamma_{11}(t)$  and  $C_{11}^{(1)} = C_{11}^{(1)}(t)$  such that if  $\mathcal{F}_1 \subseteq \binom{[n]}{v(t)}$  and  $\mathcal{F}_2 \subseteq \binom{[n]}{v(t/2)}$  satisfy  $|\mathcal{F}_1| \leq \gamma_{11}n^{v(t)}$  and  $|\mathcal{F}_2| \leq \gamma_{11}n^{v(t/2)}$ , then  $H$  is  $(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ -Ramsey with respect to  $(\mathcal{F}_1, \mathcal{F}_2)$ , whenever  $p = p(n) \geq C_{11}^{(1)}n^{-1/M_{t,t/2}}$ .*
- (P.2) *For every fixed  $\mu > 0$ , there exists a constant  $C_{11}^{(2)} = C_{11}^{(2)}(\mu, t)$  such that  $H$  is  $(\tilde{K}_{t-1}^{(3)}, \mu)$ -Ramsey, whenever  $p = p(n) \geq C_{11}^{(2)}n^{-1/M_{t-1}}$ .*
- (P.3) *For every fixed  $\mu > 0$ , there exists a constant  $C_{11}^{(3)} = C_{11}^{(3)}(\mu, t)$  such that  $H$  is  $(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)}, \mu)$ -Ramsey, whenever  $p = p(n) \geq C_{11}^{(3)}n^{-1/M_{t,t/2}}$ .*

► **Remark 12.** *A straightforward albeit somewhat tedious calculation shows that  $M_{t,t/2} \geq M_{t-1}$  holds for every even integer  $t \geq 4$ . It thus follows that Properties (P.1) and (P.3) are the most stringent in terms of the bound these impose on  $p$ . Hence, if  $p = p(n) \geq \max\{C_{11}^{(1)}, C_{11}^{(3)}\} \cdot n^{-1/M_{t,t/2}}$ , then a.a.s.  $H$  satisfies Properties (P.1), (P.2), and (P.3) simultaneously.*

Property (P.1) is modelled after [20, Theorem 2.10(i)]; Properties (P.2) and (P.3) are both specific instantiations of [20, Theorem 2.10(ii)]. The aforementioned results of [20] handle 2-graphs only. Nevertheless, proofs of Properties (P.1-3) can be attained by straightforwardly adjusting the proofs of their aforementioned counterparts in [20, Theorem 2.10] so as to accommodate the transition from 2-graphs to hypergraphs. Theorem 2.10 in [20] requires that the maximal 2-densities of the two (fixed) configurations would both be at least one; this can be omitted in our setting. Indeed, this condition is imposed in [20, Theorem 2.10] in order to handle setting (a) in that theorem where the maximal 2-densities of the two configurations coincide; by (8), this is not an issue in our case. The fact that  $\tilde{K}_t^{(3)}$  is strictly balanced with respect to  $m_3(\cdot, \tilde{K}_{t/2}^{(3)})$  is required by setting (b) appearing in [20, Theorem 2.10].

### 3.2 Proof of Theorem 1

We commence our proof of Theorem 1 with a few observations facilitating our arguments; proofs of these observations are included for completeness.

► **Observation 13.** *Let  $d \in (0, 1]$ , let  $G = (A \cup B, E)$  be a bipartite graph satisfying  $e(G) \geq d|A||B|$ , and let  $k \leq d|B|/2$  be a positive integer. Then,  $|\{v \in A : \deg_G(v) \geq k\}| \geq d|A|/2$ .*

**Proof.** Let  $A_k = \{v \in A : \deg_G(v) \geq k\}$  and suppose for a contradiction that  $|A_k| < d|A|/2$ . Then,

$$e(G) < k|A| + |A_k||B| < d|A||B|/2 + d|A||B|/2 \leq e(G)$$

which is clearly a contradiction. ◀

The next lemma captures the phenomenon of *supersaturation* (first <sup>4</sup> recorded in [22, 23, 24]) for bipartite graphs; to facilitate future references, we phrase this lemma with the host graph being bipartite as well.

► **Lemma 14.** *For every bipartite graph  $K$  and every  $d \in (0, 1)$ , there exists a constant  $\zeta = \zeta_{14} > 0$  and a positive integer  $n_0$  such that every  $n$ -vertex bipartite graph  $G = (A \cup B, E)$  satisfying  $n \geq n_0$ ,  $|A| \leq |B| \leq |A| + 1$ , and  $e(G) \geq d|A||B|$  contains at least  $\zeta n^{v(K)}$  distinct copies of  $K$ .*

► **Observation 15.** *For every graph  $K$  and every  $d \in (0, 1)$ , there exists a constant  $\xi = \xi_{15} > 0$  and an integer  $n_0$  such that the following holds whenever  $n \geq n_0$ . If an  $n$ -vertex graph  $G$  contains  $dn^{v(K)}$  distinct copies of  $K$ , then it contains at least  $\xi n$  pairwise vertex-disjoint copies of  $K$ .*

**Proof.** Any given copy of  $K$  meets  $O(n^{v(K)-1})$  copies of  $K$ . ◀

**Proof of Theorem 1.** Given  $d, t$ , and  $H$  as in the premise of Theorem 1, set

$$0 < d_3 \ll d \text{ and } 0 < \varepsilon \ll \min \left\{ d_3^{v(t/2)}, \gamma_{11}(t) \right\}. \quad (9)$$

The Tuple Property (Theorem 3) applied with  $t_3 = v(t/2)$ ,  $\varepsilon_3 = \varepsilon$ , and  $d_3^{(3)} = d_3$ , yields the existence of a constant

$$0 < \delta_3 = \delta_3^{(3)}(v(t/2), \varepsilon, d_3) \ll d_3 \quad (10)$$

as well as the functions

$$\tilde{\delta}_2(x) = \delta_3^{(2)}(x, t_3, \varepsilon, d_3, \delta_3) \text{ and } r(x) = r_3(x, t_3, \varepsilon, d_3, \delta_3),$$

where  $\tilde{\delta}_2 : \mathbb{R} \rightarrow (0, 1]$  and  $r : \mathbb{N} \rightarrow \mathbb{N}$ . Define  $\delta_2 : \mathbb{N} \rightarrow (0, 1]$  such that

$$0 < \delta_2(x) \ll \min \left\{ \tilde{\delta}_2(x), \frac{d_3^{2v(t/2)}}{v(t/2) \cdot x^{6 \cdot (2v(t/2)+1)}} \right\} \quad (11)$$

holds for every  $x \in \mathbb{N}$ . Lemma 4, applied with

$$H_1 = \dots = H_s = H, \delta_4^{(3)} = \delta_3, \delta_4^{(2)} = \delta_2, r_4 = r^5, \ell_4 \gg d_3^{-1}, \text{ and } t_4 \gg d^{-1}, \quad (12)$$

yields the existence of constants  $T_4, \tilde{t}, \ell \in \mathbb{N}$  satisfying  $t_4 \leq \tilde{t} \leq T_4$  and  $\ell_4 \leq \ell \leq T_4$ , along with partitions  $\mathcal{V} = V_1 \cup \dots \cup V_{\tilde{t}} = V(H)$  and  $(\mathcal{P}^{ij})_{1 \leq i < j \leq \tilde{t}}$  satisfying Properties (R.1-4). Set auxiliary constants

$$d_2 = 1/\ell \quad \text{and} \quad \eta = \frac{d_3^{v(t/2)} d_2^{2v(t/2)+1}}{2} \quad (13)$$

and fix

$$0 < \mu \ll \frac{\xi_{15}(\zeta_{14}(\eta/2)) \cdot d_3^{3+2v(t/2)} \cdot d_2^{10+4v(t/2)}}{v(t/2)^2 \cdot T_4}. \quad (14)$$

<sup>4</sup> Rademacher (1941, unpublished) was first to prove that every  $n$ -vertex graph with  $\lfloor n^2/4 \rfloor + 1$  edges contains at least  $\lfloor n/2 \rfloor$  triangles

## 59:12 Ramsey Properties of Randomly Perturbed Hypergraphs

We claim that there exist three distinct clusters  $X, Y, Z \in \mathcal{V}$  along with a  $(\delta_2(\ell), d_2)$ -triad  $P = P_{\alpha\beta\gamma}^{ijk}$ , with  $i, j, k, \alpha, \beta, \gamma$  appropriately defined, satisfying  $V(P) = X \cup Y \cup Z$  such that  $H[X \cup Y \cup Z]$  is  $\delta_2(\ell)$ -weakly-regular and, moreover,  $H[X \cup Y \cup Z]$  is  $(\delta_3, d_3, r)$ -regular with respect to  $P$ . To see this, note first that at most  $\tilde{t}^{\binom{n/\tilde{t}}{3}} \leq \frac{n^3}{\tilde{t}^2} \ll dn^3$  edges of  $H$  reside within the members of  $\mathcal{V}$ , where the last inequality relies on  $\tilde{t} \geq t_4 \gg d^{-1}$ , supported by (12). Second, by Property (R.3), the number of edges of  $H$  captured within  $\delta_2(\ell)$ -weakly-irregular triples  $(V_i, V_j, V_k)$ , where  $i, j, k \in [\tilde{t}]$ , is at most  $\delta_2(\ell) \cdot \tilde{t}^3 \cdot \left(\frac{n}{\tilde{t}} + 1\right)^3 \leq 2\delta_2(\ell)n^3 \ll dn^3$ , where the last inequality holds by (9) and (11). Third, by Property (R.4), the number of edges of  $H$  residing<sup>6</sup> in  $(\delta_3, d(H|P_{\alpha\beta\gamma}^{ijk}), r)$ -irregular triads  $P_{\alpha\beta\gamma}^{ijk}$  is at most  $\delta_3 n^3 \ll dn^3$ , where the last inequality holds by (9) and (10). Fourth and lastly, it follows by the Triangle Counting Lemma (Lemma 5) and by (6), that the number of edges of  $H$  found in  $(\delta_2(\ell), d_2)$ -triads  $P_{\alpha\beta\gamma}^{ijk}$ , where  $i, j, k \in [\tilde{t}]$  and  $\alpha, \beta, \gamma \in [\ell]$ , satisfying  $d(H|P_{\alpha\beta\gamma}^{ijk}) < d_3$  is at most

$$\tilde{t}^3 \ell^3 d_3 (d_2^3 + 4\delta_2(\ell)) \left(\frac{n}{\tilde{t}} + 1\right)^3 \leq 2d_3 (\ell^3 d_2^3 + 4\ell^3 \delta_2(\ell)) n^3 \stackrel{(13)}{=} (2 + 8\ell^3 \delta_2(\ell)) d_3 n^3 \ll dn^3,$$

where the last inequality holds by (9) and (11).

It follows that at least  $dn^3/2$  edges of  $H$  are captured in  $(\delta_2(\ell), d_2)$ -triads with respect to which  $H$  is  $(\delta_3, d_3, r)$ -regular and such that  $H$  is  $\delta_2(\ell)$ -weakly-regular with respect to the three members of  $\mathcal{V}$  defining the vertex-sets of these triads. The existence of  $X, Y, Z \in \mathcal{V}$  and  $P$  as defined above is then established. Throughout the remainder of the proof, we **identify**  $H$  with  $H[X \cup Y \cup Z]$ .

Let  $\mathcal{F} \subseteq \binom{X}{v(t/2)}$  be the family of all sets  $\{x_1, \dots, x_{v(t/2)}\} \subseteq X$  satisfying

$$\left| \bigcap_{j \in [v(t/2)]} L_H(x_j, P) \right| < \left( d_3^{v(t/2)} - \varepsilon \right) d_2^{2v(t/2)+1} |Y||Z|. \quad (15)$$

Then,

$$|\mathcal{F}| \leq \varepsilon |X|^{v(t/2)} \stackrel{(9)}{\ll} \gamma_{11}(t) |X|^{v(t/2)}$$

holds by (3). This application of the Tuple Lemma is supported by our choice  $\ell_4 \gg d_3^{-1}$ , seen in (12), ensuring that  $d_2 \ll d_3$  holds and thus fitting the quantification of the Tuple Lemma. With foresight (see (C.1) and (C.2) below), let

$$C = \max \left\{ C_{11}^{(1)}(t), C_{11}^{(2)}(\mu, t), C_{11}^{(3)}(\mu, t) \right\} \cdot \tilde{t}^{-1/M_{t,t/2}}$$

and put

$$p = p(n) = C \max \left\{ n^{-1/M_{t,t/2}}, n^{-1/M_{t-1}} \right\} = C n^{-1/M_{t,t/2}};$$

for the last equality consult Remark 12. Proposition 11 then asserts that the following properties are all satisfied simultaneously a.a.s. whenever  $R \sim \mathbb{H}^{(3)}(n, p)$ ; in the following list of properties, whenever an asymmetric Ramsey property is stated, the first colour is assumed to be red and the second colour is assumed to be blue.

<sup>6</sup> Supported by triangles of such triads.



- (C.1)  $R[X]$  is  $(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ -Ramsey with respect to  $(\emptyset, \mathcal{F})$ ;  
(C.2)  $R[X]$  is  $(\tilde{K}_{t/2}^{(3)}, \tilde{K}_t^{(3)})$ -Ramsey with respect to  $(\mathcal{F}, \emptyset)$ ;  
(C.3)  $R$  is  $(\tilde{K}_{t-1}^{(3)}, \mu)$ -Ramsey;  
(C.4)  $R$  is  $(\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)}, \mu)$ -Ramsey;  
(C.5)  $R$  is  $(\tilde{K}_{t/2}^{(3)}, \tilde{K}_t^{(3)}, \mu)$ -Ramsey.

Fix  $R \sim \mathbb{H}^{(3)}(n, p)$  satisfying Properties (C.1-5) and set  $\Gamma = H \cup R$ .

Let  $\psi$  be a red/blue colouring of  $E(\Gamma)$  and suppose for a contradiction that  $\psi$  does not yield any monochromatic copy of  $\tilde{K}_t^{(3)}$ . For every  $v \in V(H)$ , let  $L_H^{(r)}(v)$  denote the *red link graph of  $v$  in  $H$  under  $\psi$* , that is,  $L_H^{(r)}(v)$  is a spanning subgraph of  $L_H(v)$  consisting of the edges of  $L_H(v)$  that together with  $v$  yield a red edge of  $H$  under  $\psi$ . Similarly, let  $L_H^{(b)}(v)$  denote the *blue link graph of  $v$  in  $H$  under  $\psi$* . Note that, for any fixed vertex  $v$ , these two link subgraphs are edge-disjoint.

We say that blue (respectively, red) is a *majority colour* of  $\psi$  in  $H$  if  $|\{e \in E(H) : \psi(e) \text{ is blue}\}| \geq |\{e \in E(H) : \psi(e) \text{ is red}\}|$  (respectively,  $|\{e \in E(H) : \psi(e) \text{ is red}\}| \geq |\{e \in E(H) : \psi(e) \text{ is blue}\}|$ ).

▷ **Claim 16.** If blue is a majority colour of  $\psi$  in  $H$ , then  $e\left(L_H^{(r)}(v)\right) \leq \frac{\eta}{2v(t/2)} \cdot |Y||Z|$  holds for every  $v \in X$ .

*Proof.* Suppose for a contradiction that there exists a vertex  $v \in X$  which violates the assertion of the claim. The Triangle Counting Lemma (Lemma 5) coupled with the assumption of  $H$  being  $(\delta_3, d_3, r)$ -regular with respect to the  $(\delta_2(\ell), d_2)$ -triad  $P$  (take  $Q_1 = \dots = Q_r = P$  in (7)) collectively yield

$$\begin{aligned}
e(H) &\geq (d_3 - \delta_3)|\mathcal{K}_3(P)| \\
&\stackrel{(4)}{\geq} (d_3 - \delta_3)(d_2^3 - 4\delta_2(\ell))|X||Y||Z| \\
&\geq (d_3d_2^3 - \delta_3d_2^3 - 4d_3\delta_2(\ell))|X||Y||Z| \\
&\geq \frac{d_3d_2^3}{2}|X||Y||Z|,
\end{aligned} \tag{16}$$

where the last inequality is owing to  $\delta_3 \ll d_3$  and  $\delta_2(\ell) \ll d_2^3$  supported by (10) and (11), respectively. Blue being the majority colour implies that at least  $\frac{d_3d_2^3}{4}|X||Y||Z|$  of the edges of  $H$  are blue and thus there exists a vertex  $u \in Z$  satisfying  $e\left(L_H^{(b)}(u)\right) \geq \frac{d_3d_2^3}{4}|X||Y|$ ; note that  $L_H^{(b)}(u) \subseteq X \times Y$ . Set

$$A_v = \left\{z \in Z : \deg_{L_H^{(r)}(v)}(z) \geq t\right\} \subseteq Z \quad \text{and} \quad A_u = \left\{x \in X : \deg_{L_H^{(b)}(u)}(x) \geq t\right\} \subseteq X.$$

Then,

$$|A_v| \geq \frac{\eta}{4v(t/2)}|Z| \stackrel{(11)}{\geq} \delta_2(\ell)|Z| \quad \text{and} \quad |A_u| \geq \frac{d_3d_2^3}{8}|X| \stackrel{(11)}{\geq} \delta_2(\ell)|X| \tag{17}$$

both hold by Observation 13. Since  $H$  is  $\delta_2(\ell)$ -weakly-regular, it follows that

$$\begin{aligned}
e_H(A_u, Y, A_v) &\stackrel{(16)}{\geq} \left(\frac{d_3d_2^3}{2}\right) \cdot |A_u||Y||A_v| - \delta_2(\ell)|X||Y||Z| \\
&\stackrel{(17)}{\geq} \left(\frac{d_3d_2^3}{2}\right) \cdot \left(\frac{\eta}{4v(t/2)}\right) \cdot \left(\frac{d_3d_2^3}{8}\right) |X||Y||Z| - \delta_2(\ell)|X||Y||Z|
\end{aligned}$$

59:14 Ramsey Properties of Randomly Perturbed Hypergraphs

$$\begin{aligned}
 &= \left( \frac{d_3^2 d_2^6 \eta}{64v(t/2)} - \delta_2(\ell) \right) \cdot |X||Y||Z| \\
 &\stackrel{(11)}{\geq} \left( \frac{d_3^2 d_2^6 \eta}{65v(t/2)} \right) \cdot |X||Y||Z|. \tag{18}
 \end{aligned}$$

If red is a majority colour seen along  $E_H(A_u, Y, A_v)$ , then there exists a vertex  $v' \in A_v \subseteq Z$  satisfying

$$\left| E \left( L_H^{(r)}(v') \right) \cap (A_u \times Y) \right| \stackrel{(18)}{\geq} \left( \frac{d_3^2 d_2^6 \eta}{130v(t/2)} \right) |X||Y| \geq \left( \frac{d_3^2 d_2^6 \eta}{130v(t/2)} \right) |A_u||Y|.$$

Consequently, the set

$$A_{u,v'} = \left\{ x \in A_u : \deg_{L_H^{(r)}(v')}(x) \geq t \right\} \subseteq A_u \subseteq X$$

satisfies

$$\begin{aligned}
 |A_{u,v'}| &\geq \left( \frac{d_3^2 d_2^6 \eta}{260v(t/2)} \right) |A_u| \\
 &\stackrel{(17)}{\geq} \left( \frac{d_3^2 d_2^6 \eta}{260v(t/2)} \right) \cdot \left( \frac{d_3 d_2^3}{8} \right) |X| \\
 &\geq \left( \frac{d_3^3 d_2^9 \eta}{2100v(t/2)} \right) \cdot \left\lfloor \frac{n}{t} \right\rfloor \\
 &\stackrel{(14)}{\geq} \mu n,
 \end{aligned}$$

where the first inequality holds by Observation 13. We may then write that  $\Gamma[A_{u,v'}] \longrightarrow (\tilde{K}_{t-1}^{(3)})_2$  owing to  $R$  being  $(\tilde{K}_{t-1}^{(3)}, \mu)$ -Ramsey, by Property (C.3). Let  $K$  be a copy of  $\tilde{K}_{t-1}^{(3)}$  appearing monochromatically under  $\psi$  within  $\Gamma[A_{u,v'}]$ . Let  $x_1, \dots, x_{t-1}$  denote the branch vertices of  $K$ . It follows by the definition of  $A_{u,v'}$  that there are distinct vertices  $y_1, \dots, y_{t-1} \in Y$  such that  $\{x_i, y_i, v'\}$  is a red edge of  $H$  for every  $i \in [t-1]$ . Similarly, since  $A_{u,v'} \subseteq A_u$ , there are distinct vertices  $y'_1, \dots, y'_{t-1} \in Y$  such that  $\{x_i, y'_i, u\}$  is a blue edge of  $H$  for every  $i \in [t-1]$ . Therefore, if  $K$  is red, then it can be extended into a red copy of  $\tilde{K}_t^{(3)}$  including  $v'$ ; if, on the other hand,  $K$  is blue, then it can be extended into a blue copy of  $\tilde{K}_t^{(3)}$  including  $u$ . In either case, a contradiction to the assumption that  $\psi$  admits no monochromatic copies of  $\tilde{K}_t^{(3)}$  is reached.

It remains to consider the complementary case where blue is a majority colour in  $E_H(A_u, Y, A_v)$ . The argument in this case parallels that seen in the previous one with the sole cardinal difference being that instead of finding a monochromatic copy of  $\tilde{K}_{t-1}^{(3)}$  in a subset of  $A_u \subseteq X$ , such a copy is found in a subset of  $A_v \subseteq Z$ . An argument for this case is provided for completeness. If blue is a majority colour seen along  $E_H(A_u, Y, A_v)$ , then there exists a vertex  $u' \in A_u \subseteq X$  satisfying

$$\left| E \left( L_H^{(b)}(u') \right) \cap (Y \times A_v) \right| \stackrel{(18)}{\geq} \left( \frac{d_3^2 d_2^6 \eta}{130v(t/2)} \right) |Y||Z| \geq \left( \frac{d_3^2 d_2^6 \eta}{130v(t/2)} \right) |Y||A_v|.$$

Consequently, the set

$$A_{v,u'} = \left\{ z \in A_v : \deg_{L_H^{(b)}(u')}(z) \geq t \right\} \subseteq A_v \subseteq Z$$

satisfies

$$|A_{v,u'}| \geq \left( \frac{d_3^2 d_2^6 \eta}{260v(t/2)} \right) |A_v|$$

$$\begin{aligned}
&\stackrel{(17)}{\geq} \left( \frac{d_3^2 d_2^6 \eta}{260v(t/2)} \right) \cdot \left( \frac{\eta}{4v(t/2)} \right) |Z| \\
&\geq \left( \frac{d_3^2 d_2^6 \eta^2}{1100v(t/2)^2} \right) \cdot \left\lfloor \frac{n}{t} \right\rfloor \\
&\stackrel{(14)}{\geq} \mu n,
\end{aligned}$$

where the first inequality holds by Observation 13. Then,  $\Gamma[A_{v,u'}] \rightarrow (\tilde{K}_{t-1}^{(3)})_2$  owing to  $R$  being  $(\tilde{K}_{t-1}^{(3)}, \mu)$ -Ramsey, by Property (C.3). A monochromatic copy of  $\tilde{K}_{t-1}^{(3)}$  appearing in  $\Gamma[A_{v,u'}]$  can be either extended into a red copy of  $\tilde{K}_t^{(3)}$  including the vertex  $v$  or into a blue such copy including  $u'$ . In either case, a contradiction to the assumption that  $\psi$  admits no monochromatic copy of  $\tilde{K}_t^{(3)}$  is reached.  $\triangleleft$

The following counterpart of Claim 16 holds as well.

$\triangleright$  **Claim 17.** If red is a majority colour of  $\psi$  in  $H$ , then  $e(L_H^{(b)}(v)) \leq \frac{\eta}{2v(t/2)} \cdot |Y||Z|$  holds for every  $v \in X$ .

Proceeding with the proof of Theorem 1, assume first that blue is a majority colour of  $\psi$  in  $H$ . By Property (C.1), either there is a red copy of  $\tilde{K}_t^{(3)}$  (within  $X$ ) or there is a blue copy of  $\tilde{K}_{t/2}^{(3)}$  within  $X$  not supported on  $\mathcal{F}$ . If the former occurs, then the proof concludes. Assume then that  $K \subseteq \Gamma[X]$  is a blue copy of  $\tilde{K}_{t/2}^{(3)}$  such that  $V(K) \notin \mathcal{F}$ , and write  $L_H(K, P) = \bigcap_{x \in V(K)} L_H(x, P)$  to denote the joint link graph of the members of  $V(K)$  supported on  $P$ . Then,

$$e(L_H(K, P)) \geq \left( d_3^{v(t/2)} - \varepsilon \right) d_2^{2v(t/2)+1} |Y||Z|,$$

holds by (15). Remove  $E(L_H^{(r)}(x))$  from  $E(L_H(K, P))$  for every  $x \in V(K)$ ; that is, remove any edge in  $L_H(K, P)$  that together with a vertex of  $K$  gives rise to a red edge of  $H$  with respect to  $\psi$ . By Claim 16, at most

$$\sum_{x \in V(K)} e(L_H^{(r)}(x)) \leq v(t/2) \cdot \frac{\eta}{2v(t/2)} |Y||Z| = \frac{\eta}{2} |Y||Z|$$

edges are thus discarded from  $L_H(K, P)$ , leaving at least

$$\begin{aligned}
\left[ \left( d_3^{v(t/2)} - \varepsilon \right) d_2^{2v(t/2)+1} - \frac{\eta}{2} \right] |Y||Z| &\stackrel{(9)}{\geq} \left( \frac{d_3^{v(t/2)} d_2^{2v(t/2)+1}}{2} - \frac{\eta}{2} \right) |Y||Z| \\
&\stackrel{(13)}{=} \left( \eta - \frac{\eta}{2} \right) |Y||Z| \\
&= \frac{\eta}{2} |Y||Z|
\end{aligned}$$

edges in the *residual* joint link graph of  $K$ , denoted  $L'_H(K, P)$ . It follows by Lemma 14 and Observation 15 that  $L'_H(K, P)$  contains at least

$$\xi_{15}(\zeta_{14}(\eta/2)) \frac{2n}{T_4} \stackrel{(14)}{\geq} \mu n$$

vertex-disjoint copies of the bipartite graph  $K_{1,t/2}$ . Let  $S \subseteq V(L'_H(K, P))$  consist of the centre-vertices of all said copies of  $K_{1,t/2}$ . Property (C.4) coupled with  $|S| \geq \mu n$  collectively assert that  $\Gamma[S] \rightarrow (\tilde{K}_t^{(3)}, \tilde{K}_{t/2}^{(3)})$ . If the first alternative occurs, then there is a red copy

of  $\tilde{K}_t^{(3)}$  and thus the proof concludes. Suppose then that the second alternative takes place so that a blue copy  $K'$  of  $\tilde{K}_{t/2}^{(3)}$  arises in  $\Gamma[S]$ . Let  $u_1, \dots, u_{t/2}$  denote the branch-vertices of  $K'$  and let  $x_1, \dots, x_{t/2}$  denote the branch-vertices of  $K$ . It follows by the definitions of  $L'_H(K, P)$  and  $S$  that there are  $t^2/4$  distinct vertices  $\{w_{ij} : i, j \in [t/2]\} \subseteq V(L'_H(K, P)) \setminus \{u_1, \dots, u_{t/2}, x_1, \dots, x_{t/2}\}$  such that  $\{u_i, x_j, w_{ij}\}$  forms a blue edge of  $H$  for every  $i, j \in [t/2]$ . We conclude that  $\Gamma$  admits a copy of  $\tilde{K}_t^{(3)}$  which is blue under  $\psi$ .

Next, assume that red is a majority colour seen for  $\psi$  in  $H$ . Replacing the appeals to Claim 16, Properties (C.1) and (C.4) in the argument above with appeals to Claim 17 and Properties (C.2), and (C.5), respectively, leads to the rise of a monochromatic copy of  $\tilde{K}_t^{(3)}$  in  $\Gamma$  under  $\psi$  in this case as well.  $\blacktriangleleft$

---

## References

- 1 E. Aigner-Horev, O. Danon, D. Hefetz, and S. Letzter. Large rainbow cliques in randomly perturbed dense graphs. *SIAM J. Discrete Math.*, 36(4):2975–2994, 2022.
- 2 E. Aigner-Horev, O. Danon, D. Hefetz, and S. Letzter. Small rainbow cliques in randomly perturbed dense graphs. *European J. Combin.*, 101:Paper No. 103452, 34 pages, 2022.
- 3 E. Aigner-Horev and D. Hefetz. Rainbow hamilton cycles in randomly colored randomly perturbed dense graphs. *SIAM J. Discrete Math.*, 35(3):1569–1577, 2021.
- 4 E. Aigner-Horev, D. Hefetz, and M. Krivelevich. Minors, connectivity, and diameter in randomly perturbed sparse graphs. Arxiv preprint, 2022. [arXiv:2212.07192](#).
- 5 E. Aigner-Horev, D. Hefetz, and M. Krivelevich. Cycle lengths in randomly perturbed graphs. *Random Structures & Algorithms*, 63(4):867–884, 2023.
- 6 E. Aigner-Horev, D. Hefetz, and A. Lahiri. Rainbow trees in uniformly edge-colored graphs. *Random Structures & Algorithms*, 62(2):287–303, 2023.
- 7 E. Aigner-Horev, D. Hefetz, and M. Schacht. Ramsey properties of randomly perturbed hypergraphs. Arxiv preprint, 2024. [arXiv:2311.01750](#).
- 8 E. Aigner-Horev and Y. Person. Monochromatic Schur triples in randomly perturbed dense sets of integers. *SIAM J. Discrete Math.*, 33(4):2175–2180, 2019.
- 9 P. Allen, O. Parczyk, and V. Pfenninger. Resilience for tight Hamiltonicity. Arxiv preprint, 2021. [arXiv:2105.04513](#).
- 10 J. Balogh, A. Treglown, and A. Z. Wagner. Tilings in randomly perturbed dense graphs. *Combin. Probab. Comput.*, 28(2):159–176, 2019.
- 11 W. Bedenkecht, J. Han, Y. Kohayakawa, and G. O. Mota. Powers of tight hamilton cycles in randomly perturbed hypergraphs. *Random Structures & Algorithms*, 55(4):795–807, 2019.
- 12 J. Böttcher, J. Han, Y. Kohayakawa, R. Montgomery, O. Parczyk, and Y. Person. Universality for bounded degree spanning trees in randomly perturbed graphs. *Random Structures & Algorithms*, 55(4):854–864, 2019.
- 13 J. Böttcher, R. Montgomery, O. Parczyk, and Y. Person. Embedding spanning bounded degree graphs in randomly perturbed graphs. *Mathematika*, 66(2):422–447, 2020.
- 14 C. Bowtell, R. Hancock, and J. Hyde. Proof of the Kohayakawa–Kreuter conjecture for the majority of cases. Arxiv preprint, 2023. [arXiv:2307.16760](#).
- 15 M. Christoph, A. Martinsson, R. Steiner, and Y. Wigderson. Resolution of the Kohayakawa–Kreuter conjecture. Arxiv preprint, 2024. [arXiv:2402.03045](#).
- 16 D. Conlon, J. Fox, and V. Rödl. Hedgehogs are not colour blind. *J. Combinatorics*, 8(3):475–485, 2017.
- 17 D. Conlon and W. T. Gowers. Combinatorial theorems in sparse random sets. *Ann. of Math.*, 184(2):367–454, 2016.
- 18 S. Das, C. Knierim, and P. Morris. Schur properties of randomly perturbed sets. Arxiv preprint, 2022. [arXiv:2205.01456](#).
- 19 S. Das, P. Morris, and A. Treglown. Vertex Ramsey properties of randomly perturbed graphs. *Random Structures & Algorithms*, 57(4):983–1006, 2020.

- 20 S. Das and A. Treglown. Ramsey properties of randomly perturbed graphs: cliques and cycles. *Combin. Probab. Comput.*, 29(6):830–867, 2020.
- 21 A. Dudek, C. Reiher, A. Ruciński, and M. Schacht. Powers of Hamiltonian cycles in randomly augmented graphs. *Random Structures & Algorithms*, 56(1):122–141, 2020.
- 22 P. Erdős. Some theorems on graphs. *Riveon Lematematika*, 9:13–17, 1955.
- 23 P. Erdős. On a theorem of rademacher-turán. *Illinois J. Math.*, 6:122–127, 1962.
- 24 P. Erdős. On the number of complete subgraphs contained in certain graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 7:459–464, 1962.
- 25 P. Frankl and V. Rödl. Extremal problems on set systems. *Random Structures & Algorithms*, 20(2):131–164, 2002.
- 26 E. Friedgut, V. Rödl, and M. Schacht. Ramsey properties of random discrete structures. *Random Structures & Algorithms*, 37(4):407–436, 2010.
- 27 L. Gugelmann, R. Nenadov, Y. Person, N. Škorić, A. Steger, and H. Thomas. Symmetric and asymmetric ramsey properties in random hypergraphs. *Forum Math. Sigma*, 5:Paper No. e28, 47 pages, 2017.
- 28 J. Han and Y. Zhao. Hamiltonicity in randomly perturbed hypergraphs. *J. Combin. Theory Ser. B*, 144:14–31, 2020.
- 29 J. Hyde. Towards the 0-statement of the Kohayakawa–Kreuter conjecture. Arxiv preprint, 2021. [arXiv:2105.15151](https://arxiv.org/abs/2105.15151).
- 30 Y. Kohayakawa and B. Kreuter. Threshold functions for asymmetric ramsey properties involving cycles. *Random Structures & Algorithms*, 11(3):245–276, 1997.
- 31 Y. Kohayakawa, M. Schacht, and R. Spöhel. Upper bounds on probability thresholds for asymmetric Ramsey properties. *Random Structures & Algorithms*, 44(1):1–28, 2014.
- 32 J. Komlós, M. Shokoufandeh, A. Simonovits, and E. Szemerédi. The regularity lemma and its applications in graph theory. In *Theoretical aspects of computer science*, volume 2292 of *Lecture Notes in Comput. Sci.*, pages 84–112. Springer, Berlin, 2002.
- 33 J. Komlós and M. Simonovits. Szemerédi’s regularity lemma and its applications in graph theory. In *Combinatorics, Paul Erdős is eighty, Vol. 2*, volume 2 of *Bolyai Soc. Math. Stud.* János Bolyai Math. Soc., Budapest, 1996.
- 34 M. Krivelevich, M. Kwan, and B. Sudakov. Cycles and matchings in randomly perturbed digraphs and hypergraphs. *Combin. Probab. Comput.*, 25(6):909–927, 2016.
- 35 M. Krivelevich, M. Kwan, and B. Sudakov. Bounded-degree spanning trees in randomly perturbed graphs. *SIAM J. Discrete Math.*, 31(1), 2017.
- 36 M. Krivelevich, B. Sudakov, and Prasad Tetali. On smoothed analysis in dense graphs and formulas. *Random Structures & Algorithms*, 29(2):180–193, 2006.
- 37 E. Kuperwasser, W. Samotij, and Y. Wigderson. On the kohayakawa–kreuter conjecture. Arxiv preprint, 2023. [arXiv:2307.16611](https://arxiv.org/abs/2307.16611).
- 38 A. Liebenau, L. Mattos, W. Mendonça, and J. Skokan. Asymmetric Ramsey properties of random graphs involving cliques and cycles. *Random Structures & Algorithms*, 62(4):1035–1055, 2023.
- 39 T. Łuczak, A. Ruciński, and B. Voigt. Ramsey properties of random graphs. *J. Combin. Theory Ser. B*, 56(1):55–68, 1992.
- 40 M. Marciniszyn, J. Skokan, , R. Spöhel, and A. Steger. Asymmetric Ramsey properties of random graphs involving cliques. *Random Structures & Algorithms*, 34(4):419–453, 2009.
- 41 A. McDowell and R. Mycroft. Hamilton  $\ell$ -cycles in randomly perturbed hypergraphs. *Electron. J. Combin.*, 25(4):Paper No. 4.36, 30 pages, 2018.
- 42 F. Mousset, R. Nenadov, and W. Samotij. Towards the Kohayakawa–Kreuter conjecture on asymmetric Ramsey properties. *Combin. Probab. Comput.*, 29(6):943–955, 2020.
- 43 D. Mubayi. A hypergraph extension of turán’s theorem. *J. Combin. Theory Ser. B*, 96(1):122–134, 2006.
- 44 R. Nenadov, Y. Person, N. Škorić, and A. Steger. An algorithmic framework for obtaining lower bounds for random Ramsey problems. *J. Combin. Theory Ser. B*, 124:1–38, 2017.

## 59:18 Ramsey Properties of Randomly Perturbed Hypergraphs

- 45 Rajko. Nenadov and A. Steger. A short proof of the random Ramsey theorem. *Combin. Probab. Comput.*, 25(1):130–144, 2016.
- 46 E. Powierski. Ramsey properties of randomly perturbed dense graphs. Arxiv preprint, 2019. [arXiv:1902.02197](https://arxiv.org/abs/1902.02197).
- 47 V. Rödl and A. Ruciński. Lower bounds on probability thresholds for ramsey properties. In *Combinatorics, Paul Erdős is eighty, Vol. 1*, Bolyai Soc. Math. Stud., pages 317–346. János Bolyai Math. Soc., Budapest, 1993.
- 48 V. Rödl and A. Ruciński. Random graphs with monochromatic triangles in every edge coloring. *Random Structures & Algorithms*, 5(2):253–270, 1994.
- 49 V. Rödl and A. Ruciński. Threshold functions for Ramsey properties. *J. Amer. Math. Soc.*, 8(4):917–942, 1995.
- 50 V. Rödl and A. Ruciński. Ramsey properties of random hypergraphs. *J. Combin. Theory Ser. A*, 81(1):1–33, 1998.
- 51 V. Rödl and M. Schacht. Regular partitions of hypergraphs: regularity lemmas. *Combin. Probab. Comput.*, 16(6):833–885, 2007.
- 52 D. Spielman and S. Teng. Smoothed Analysis of algorithms: why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51:385–463, 2004.
- 53 D. Spielman and S. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52:76–84, 2009.
- 54 E. Szemerédi. Regular partitions of graphs. In *Problèmes combinatoires et théorie des graphes*, volume 260 of *Colloq. Internat. CNRS*, pages 399–401. CNRS, Paris, 1978.

# Nearly Optimal Local Algorithms for Constructing Sparse Spanners of Clusterable Graphs

Reut Levi ✉ 

Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Moti Medina ✉ 

Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

Omer Tubul ✉

Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

---

## Abstract

In this paper, we study the problem of locally constructing a sparse spanning subgraph (LSSG), introduced by Levi, Ron, and Rubinfeld (ALGO'20). In this problem, the goal is to locally decide for each  $e \in E$  if it is in  $G'$  where  $G'$  is a connected subgraph of  $G$  (determined only by  $G$  and the randomness of the algorithm). We provide an LSSG that receives as a parameter a lower bound,  $\phi$ , on the conductance of  $G$  whose query complexity is  $\tilde{O}(\sqrt{n}/\phi^2)$ . This is almost optimal when  $\phi$  is a constant since  $\Omega(\sqrt{n})$  queries are necessary even when  $G$  is an expander. Furthermore, this improves the state of the art of  $\tilde{O}(n^{2/3})$  queries for  $\phi = \Omega(1/n^{1/12})$ .

We then extend our result for  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graphs and provide an algorithm whose query complexity is  $\tilde{O}(\sqrt{n} + \phi_{\text{out}}n)$  for constant  $k$  and  $\phi_{\text{in}}$ . This bound is almost optimal when  $\phi_{\text{out}} = O(1/\sqrt{n})$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms; Mathematics of computing  $\rightarrow$  Graph algorithms; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Locally Computable Algorithms, Sublinear algorithms, Spanning Subgraphs, Clusterable Graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.60

**Category** RANDOM

**Funding** *Reut Levi*: The author was supported by the Israel Science Foundation under Grant 1867/20.

*Moti Medina*: The author was supported by the Israel Science Foundation under Grants 867/19 and 554/23.

*Omer Tubul*: The author was supported by the Israel Science Foundation under Grants 867/19 and 554/23.

## 1 Introduction

When dealing with huge graphs, several practical constraints arise: (i) **Memory Limitations**: It is often impractical or infeasible to store the entire graph in the local memory of a processing unit. (ii) **Algorithmic Efficiency**: Due to the graph's size, running linear-time (or even slower) algorithms becomes challenging. (iii) **Parallel Computation**: Relying on a single processing unit for computations can be inefficient. The Centralized Local model, also called Locally Computable Algorithms (LCA), was introduced by Rubinfeld et al. [27] to address these challenges. This model treats the input graph as if it is stored in a (likely distributed) database. External processing units can query this database to perform computations efficiently. The system prohibits shared memory or communication between



© Reut Levi, Moti Medina, and Omer Tubul;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 60; pp. 60:1–60:21



Leibniz International Proceedings in Informatics

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

querying processes to reduce coordination overhead. Instead, shared randomness accompanies input access. The approach involves running sublinear-time algorithms to extract global graph properties or locally examining the input graph as needed by applications.

One of the problems studied in this model is locally constructing a sparse spanning subgraph of a connected input graph. It was introduced and formalized by Levi, Ron, and Rubinfeld [17] as follows.

► **Definition 1** ([17]). *An algorithm  $\mathcal{A}$  is a Local Sparse Spanning Graph (LSSG) algorithm if, given  $n \geq 1$ ,  $\varepsilon > 0$ , a sequence of random bits  $r \in \{0, 1\}^*$  and query access to the incidence-lists representation of a connected graph  $G = (V, E)$  over  $n$  vertices,<sup>1</sup> it provides oracle access to a subgraph  $G' = (V, E')$  of  $G$  such that:*

- (1)  $G'$  is connected, and
- (2)  $|E'| \leq (1 + \varepsilon) \cdot n$  with probability at least  $1 - 1/\Omega(n)$ , where  $E'$  is determined by  $G$  and  $r$ .<sup>2 3</sup>

As observed in [17], if we insist that  $G'$  should have the minimum number of edges sufficient to span  $G$ , namely, that  $G'$  be a spanning tree, then the task cannot be performed by an LCA in general without inspecting almost all of  $G$ . On the other hand, even under the above relaxation, [17] showed that this task requires  $\Omega(\sqrt{n})$  queries.<sup>4</sup> They complimented this negative result with an almost tight upper bound that works under the promise that the input graph expands extremely fast. In particular, their algorithm is tight when the growth rate of their graph is  $\Omega(d)$  for small sets (of size roughly  $\sqrt{n}$ ), where  $d$  is the maximum vertex degree. In fact, their result achieves a sublinear query complexity only when the growth rate (on small sets) is at least  $d^{1/2+1/\log n}$ .<sup>5</sup> This raises the question *whether it is possible to obtain a similar result also for graphs whose growth rate is just a small constant?* (which, in particular, may not depend on  $d$ ). Namely, for graphs, which are just good expanders. We answer this question affirmatively by showing almost tight results for expanders. Moreover, our upper bound works for general graphs and receives as a parameter a lower bound on the conductance of the graph, defined as follows.

► **Definition 2** (Graph Conductance). *Let  $G = (V, E)$  be an undirected graph with maximum vertex degree  $d$ . Let  $S \subseteq V$  denote a nonempty subset of  $V$ , then the conductance of  $S$  w.r.t.  $G$  is  $\phi_G(S) \stackrel{\text{def}}{=} \frac{e(S, V \setminus S)}{d|S|}$ , where  $e(A, B) \stackrel{\text{def}}{=} |\{\{a, b\} \in E \mid a \in A, b \in B\}|$ , for  $A, B \subseteq V$ . The conductance of  $G$ ,  $\phi(G)$  is then defined as*

$$\phi(G) \stackrel{\text{def}}{=} \min_{\substack{S \subseteq V \\ |S| \leq |V|/2}} \phi_G(S). \quad (1)$$

To explain why conductance comes into play in our algorithm, we first describe a common paradigm for constructing sparse spanning subgraphs and spanners. In many cases, the paradigm is to select a random set of vertices, also referred to as *centers*, and to partition the vertices of the graphs into Voronoi cells according to this selection of centers. Usually, it is easy to span the Voronoi cells (assuming the centers are selected u.a.r.), and the challenging

<sup>1</sup> Namely, the algorithm can query each vertex  $v \in V$  and an index  $i$ , who is the  $i$ th neighbor of  $v$  (where if  $v$  has less than  $i$  neighbors, then a special symbol is returned, e.g., “no neighbor”).

<sup>2</sup>  $E'$  is determined only by  $G$  and  $r$  and not from, e.g., the queries made to the oracle or their order.

<sup>3</sup> We say that an event occurs *with high probability* (w.h.p) if it occurs with probability at least  $1 - 1/\Omega(n)$ .

<sup>4</sup> One way to show this is by reducing the problem of testing cycle-freeness to the LSSG problem.

<sup>5</sup> To illustrate this drawback, note that even for  $d = 5$  and growth rate = 2 their algorithm is not guaranteed to be sublinear.



part is to preserve the connectivity between Voronoi cells. This is also the approach taken by [17] (and generalized in [12, 23, 2]). To bypass the challenging part of connecting the Voronoi cells, we aim to choose centers that are initially connected and distributed nearly uniformly. To this end, we select the centers by performing random walks from a single vertex and taking the endpoints of these walks to be the set of centers. We set the length of the random walks to be at least the mixing time of the graph, which, in turn, depends on the conductance of the graph, so the selected centers are distributed almost uniformly. We then add the edges traversed by the random walks to our constructed subgraph, so it only remains to span the Voronoi cells (which can be done efficiently, assuming the centers are distributed almost uniformly).

Since every graph can be partitioned into expanders (see, e.g., [22]), one may wonder if this approach can be extended to work in general graphs. To make this question concrete, consider an input graph composed of two expanders (of equal size) connected with a single edge between them. While the *inner conductance* of each expander is high, the overall conductance of the graph is  $O(1/n)$ , and consequently, the mixing time of the graph is high. So, the question that might come to mind is whether it is possible to extend the approach mentioned above so the length of the random walks will depend on the inner-conductance of the expanders (which is a constant) and not on the conductance of the entire graph. This question becomes more interesting as the connectivity between the expanders, which is referred to as the *outer-conductance* of the expanders, grows but not to the extent in which the overall conductance is high. We show that our approach can be extended to support a wide range of inner-conductance and outer-conductance. More specifically, we extend our result to clusterable graphs as defined in the work of Gharan and Trevisan [9] and Czumaj, Peng, and Sohler [6].

► **Definition 3** (Graph Clusterability. Based by [6]). *Let  $G = (V, E)$  be an undirected graph with maximum degree  $d$ . Let  $n \stackrel{\text{def}}{=} |V|$ . For any  $S \subseteq V$ , let  $G[S]$  be the induced subgraph of  $G$  on the vertex set  $S$ . We say that a graph  $G$  is  $(k, \phi)$ -clusterable, where  $k \in \{1, \dots, n\}$ ,  $\phi \in [0, 1]$ , if there exists a partition of  $V$  into  $h$  sets  $C_1, \dots, C_h$  s.t.  $1 \leq h \leq k$ , and that for each  $i \in \{1, \dots, h\}$  it holds that  $\phi(G[C_i]) \geq \phi$ . We refer to each  $C_i$  as a  $\phi$ -cluster and the corresponding partition to  $h$  clusters as an  $(h, \phi)$ -clustering. Similarly, We say that a graph  $G$  is  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable, if for each  $i \in \{1, \dots, h\}$  it holds that  $\phi(G[C_i]) \geq \phi_{\text{in}}$  and  $\phi_G(C_i) \leq \phi_{\text{out}}$ . We refer to each  $C_i$  as a  $(\phi_{\text{in}}, \phi_{\text{out}})$ -cluster.*<sup>6</sup>

Aside from connectivity, another desirable property of  $G'$  is that it will preserve the pairwise distances between vertices. In particular, we say that the subgraph  $G'$  is an  $\alpha$ -spanner of  $G$  if for every  $u, v \in V$ ,  $\text{dist}_{G'}(u, v) \leq \alpha \cdot \text{dist}_G(u, v)$  where  $\text{dist}_{G'}(u, v)$  and  $\text{dist}_G(u, v)$  denote the length of the shortest path from  $u$  to  $v$  in  $G'$  and  $G$ , respectively. We refer to  $\alpha$  as the *stretch factor* of the spanner  $G'$ .

In the next section, we state the performances of our upper bounds in terms of their query complexity, the number of random bits they use, and the stretch factor of the obtained spanning subgraph,  $G'$ .

## 1.1 Our Results

Our first result is an LSSG that receives as a parameter a lower bound,  $\phi$ , on the conductance of the graph, whose query complexity is  $\tilde{O}(\sqrt{n}/\phi^2)$  for constant  $d$ , as stated next.

<sup>6</sup> Note that requiring  $\phi > 0$  implies that each induced cluster of  $G$  is also connected.

► **Theorem 4.** *There is an LSSG algorithm that given query access to a connected graph  $G = (V, E)$  and a lower bound  $\phi$  on  $\phi(G)$ , provides access to  $G' = (V, E')$  such that the following holds. (1) The graph  $G'$  is a connected subgraph of  $G$  and with high probability  $|E'| = n + O\left(\frac{\sqrt{n}\log^2 n}{\phi^2}\right)$ . Moreover, the stretch factor of  $G'$  is  $O\left(\frac{\log n}{\phi^2}\right)$ . (2) The query complexity of the algorithm is  $O\left(\sqrt{n} \cdot \left(\frac{\log^2 n}{\phi^2} + d^2\right)\right)$ , and (3) the number of random bits it uses is  $O\left(\frac{\log d \cdot \log^2 n}{\phi^2}\right)$ , where  $d$  is a bound on the maximum degree of  $G$ .*

Therefore, for constant  $\phi$  and constant  $d$  this upper bound is tight, up to polylogarithmic factors in  $n$ . Moreover, it improves the state-of-the-art upper bound for general bounded-degree graph of  $\tilde{O}(n^{2/3})$  queries by Lenzen and Levi [12] if and only if  $\phi > n^{-1/12}$ . Consequently, we may assume that  $\phi > n^{-1/12}$  throughout this paper. As a result, we obtain that  $|E'| = n + o(n)$ , which makes  $G'$  an ultra-sparse spanner of stretch  $O(\log n/\phi^2)$ .

Our next main result is stated in the next theorem.

► **Theorem 5.** *There is an LSSG algorithm that given query access to a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph  $G = (V, E)$ , where each cluster is of size at least  $\beta \cdot n$ , provides access to  $G' = (V, E')$  such that the following holds. (1) The graph  $G'$  is a connected subgraph of  $G$  and with high probability  $|E'| \leq n(1 + \varepsilon)$ . Moreover, the stretch factor of  $G'$  is  $\Theta\left(\frac{\log n}{\phi_{\text{in}}^2}\right)$ . (2) The query complexity of the algorithm is  $O\left(\log^2 n \cdot (\beta\phi_{\text{out}})^{-1} + n \log^2 n \cdot k^3 d^3 \phi_{\text{out}} (\varepsilon\phi_{\text{in}})^{-1}\right)$ , and (3) the number of random bits it uses is  $O\left(\frac{\log d \cdot \log^2 n}{\phi_{\text{in}}^2}\right)$ , where  $d$  is a bound on the maximum degree of  $G$ .*

For instances where  $\phi_{\text{in}}, k, \beta, \varepsilon$ , and  $d$  are constants, we obtain the following corollary.

► **Corollary 6.** *There is an LSSG algorithm that given query access to a connected  $(\Theta(1), \Theta(1), \phi_{\text{out}})$ -clusterable graph  $G = (V, E)$ , where each cluster is of size at least  $\beta \cdot n$ , provides access to  $G' = (V, E')$  such that the following holds. (1) The graph  $G'$  is a connected subgraph of  $G$  and with high probability  $|E'| \leq n(1 + \varepsilon)$ . Moreover, the stretch factor of  $G'$  is  $\Theta(\log n)$ . (2) The query and time complexity of the algorithm is  $\tilde{O}(\sqrt{n} + \phi_{\text{out}} n)$ , and (3) the number of random bits it uses is  $O(\log^2 n)$ .*

Namely, our algorithm is nearly tight in this case as long as  $\phi_{\text{out}} = O(1/\sqrt{n})$  and improves over the state-of-the-art for  $\phi_{\text{out}} = O(1/n^{1/3})$ .

## 1.2 Overview of Our Algorithms

### 1.2.1 The case of a single cluster

We begin by describing our algorithm for  $k = 1$  for the case where the graph is rapidly mixing, namely, that its mixing time,  $\tau$ , is  $O(\log n)$ . We first describe the algorithm from a global point of view. The algorithm picks an arbitrary vertex as a *primary-center*. It then performs  $\tilde{\Theta}(\sqrt{n})$  lazy random walks of length  $\tau$  from that center, where  $\tau$  denotes the mixing time of the graph. The end-vertex of each one of these walks is added to the set of *secondary centers*. The edges traversed by these random walks are added to  $E'$ . Consequently, there is a path of length at most  $2\tau$  between every pair of secondary centers. In the second step, the graph's vertex set is partitioned into Voronoi-cells with respect to the selected secondary centers. Namely, each vertex joins the cell of its closest secondary center (breaking ties by ids). A spanning tree of each Voronoi-cell is then added to the spanner. The specific spanning tree added is rooted at the secondary center, where the path from each vertex to the root has the

least lexicographical order. As shown by [17], it is possible to reconstruct the edges incident to a vertex  $v$  in this tree at the same cost as performing the BFS exploration to find the secondary center of  $v$ . The resulting subgraph clearly spans the graph: for a pair of vertices  $u$  and  $v$  in the same Voronoi cell, there is a path between  $u$  and the respective secondary center of the cell and likewise for  $v$ ; If  $u$  and  $v$  are not in the same Voronoi cell, then their secondary centers are connected to the primary-center by paths of length at most  $\tau$ . Thus the stretch-factor of the spanner is  $O(\tau + \ell)$  where  $\ell$  is an upper bound on the diameter of the Voronoi-cells, which is bounded by  $O(\log n/\phi^2)$ .

The local implementation proceeds as follows. On query  $\{u, v\}$ , the local algorithm simulates the first step of the global algorithm and returns YES if  $\{u, v\}$  is an edge traversed by one of the random walks performed by the algorithm. Otherwise, it performs a BFS, layer by layer (that is, it reveals an entire layer in each step) from  $u$  until it finds the secondary center of  $u$ , likewise for  $v$ . If the centers of  $u$  and  $v$  are different, then the algorithm returns NO. Otherwise, it returns YES iff  $\{u, v\}$  is an edge of the tree selected to span the Voronoi-cell of  $u$  and  $v$  (as described above).

The query and time complexity of performing the first step of the local algorithm is clearly  $\tilde{O}(\sqrt{n} \cdot \tau)$ . For the second step, since the length of the random walks performed in the first step is  $\tau$ , the  $\Theta(\sqrt{n})$  secondary centers are distributed almost uniformly in the graph. Therefore, with high probability, each vertex in the graph sees a secondary center after exploring  $\tilde{O}(\sqrt{n})$  vertices. If this is not the case, we can afford to add all the edges incident to  $v$  to  $E'$  without harming the sparsity of  $G'$  while preserving the connectivity of  $G'$ .

### 1.3 The case of $k$ -clusterable graphs

We next describe our algorithm for  $k$ -clusterable graphs for the case that the mixing time of each cluster is  $O(\log n)$ , namely, that  $\phi_{\text{in}}$  is a constant, and  $\phi_{\text{out}}$  is  $O(1/\sqrt{n})$ . The first phase of the algorithm is similar to the algorithm for a single cluster. The only difference is that we start with  $\Theta(\log n)$  primary-centers (chosen uniformly at random) rather than a single one. Thus, with high probability we hit every cluster with at least one primary-center. Therefore, after the first phase, our spanner consists of edges of  $\tilde{\Theta}(\sqrt{n})$  random walks, traversed from each one of the primary centers, and the edges of the spanning trees of the Voronoi-cells which are constructed with respect to the set of secondary centers (which are now originated from several primary centers). Let us call the set of secondary centers originating from the same primary-center and their respective Voronoi-cells an *artificial-cluster*. Clearly, after the first step of the algorithm, the spanner spans each of the artificial clusters (from the same reasoning as above). Our goal is to ensure that every pair of artificial-clusters with an edge in their cut in the original graph will have an edge in their cut in the spanner. To this end, we sample u.a.r. a set of  $\Theta(\log n/\epsilon)$  edges and add these edges to the spanner. Let us denote this set of edges by  $\mathcal{T}$ . For every pair of artificial-clusters whose cut does not intersect the set  $\mathcal{T}$ , we add all the edges in their cut to the spanner. The rationale is that if the respective cut is large, it will intersect  $\mathcal{T}$ ; otherwise, we can afford to add its edges to the spanner.

**The analysis.** The challenging part in analyzing this algorithm is to show that the secondary centers are distributed (almost) uniformly in each one of the clusters. This is crucial for proving that the query complexity remains  $\tilde{O}(\sqrt{n})$ . More specifically, this is crucial for claiming that with high probability, each vertex sees a secondary center after exploring  $\tilde{O}(\sqrt{n})$  vertices. This is where the requirement on the outer-conductance of each cluster comes into play. We show that if the outer-conductance is  $O(1/\sqrt{n})$  then for a constant

fraction of the vertices in the cluster,  $v$ , the end-vertex of a random walk from  $v$  of length  $O(\log n)$  is likely to be any vertex in the cluster w.p. at least  $\Omega(1/n)$  except for a small set of vertices of size  $O(\sqrt{n})$ . This is sufficient to upper bound the query complexity of the BFS exploration to find the center.

## 1.4 Related Work

### 1.4.1 LSSG Algorithms

The problem of finding a sparse spanning subgraph in the Centralized Local model was first studied in [16, 17], where the authors show a lower bound of  $\Omega(\sqrt{n})$  queries for constant  $\varepsilon$  and  $\Delta$  (see also survey by Rubinfeld [26]). They also present an upper bound with nearly tight query complexity for graphs with very good expansion properties.

In [17], the authors also provide an efficient algorithm for minor-free graphs that was later improved in [19]. The algorithm presented in [19] achieves a polynomial query complexity in  $\Delta$  and  $1/\varepsilon$  and is independent of  $n$ . The stretch factor of this algorithm is also independent of  $n$  and depends only on  $\Delta$ ,  $1/\varepsilon$ , and the size of the excluded minor  $h$ . A more general family of graphs, hyperfinite graphs, is studied in [14]. They show an upper bound (which builds on an algorithm in [17]) that has a query complexity that is independent of  $n$  (however, super-exponential in  $1/\varepsilon$ ). Informally, both minor-free graphs and hyperfinite graphs are families of graphs that are, roughly speaking, sufficiently non-expanding everywhere. On the other hand, they show that, for a family of graphs with expansion properties that are slightly better, any local algorithm must have a query complexity that depends on  $n$ .

The first LSSG algorithm for general (bounded degree) graphs was introduced in [12], presenting a query complexity of  $\tilde{O}(n^{2/3} \cdot \text{poly}(1/\varepsilon, d))$  and a stretch factor of  $O(\log^2 n \cdot \text{poly}(d/\varepsilon))$ . Recently, Bodwin and Fleischmann [4] introduced an Adjacency Oracle for a Spanning Subgraph of  $(1 + \varepsilon)n$  edges for general (non-bounded degree) graphs that works in  $\tilde{O}(n/\varepsilon)$  time, hence sublinear in the number of edges on a dense graph. Adjacency Oracles are closely related to LCA, except that Adjacency Oracles are allowed to perform a centralized pre-processing but demand a query time of  $\tilde{O}(1)$ . Their Adjacency Oracle implies an LSSG algorithm for general (non-bounded degree) graphs in  $\tilde{O}(n)$  time, which works by constructing an Adjacency Oracle and uses it once for each query.

For more related work on LCAs for spanners, graph clustering, and LCAs for other graph problems, see Appendix A.

## 2 Preliminaries

In this section, we describe our main technical tools. Omitted proofs appear in Appendix B.

**Notation.** Throughout this paper, we consider a bounded degree (undirected) simple graph  $G = (V, E)$ , where  $V = [n]$  and its maximum degree is  $d = \max_{v \in V} d_G(v)$ , where  $d_G(v)$  denotes the degree of  $v$  w.r.t. the graph  $G$ . The identifier (ID in short) of a vertex  $v \in V$  is simply  $v$ . For each  $A \subseteq V$ , we define  $N_G(A)$  to be the number of neighbors of  $A$  *outside* of  $A$  in  $G$ , i.e.,  $N_G(A) \stackrel{\text{def}}{=} |\{v \in V \setminus A \mid \{u, v\} \in E, u \in A\}|$ . When the graph  $G$  is clear from the context, we omit the subscript  $G$ . For a vertex  $v$ , we call the set of vertices of distance at most  $\ell$  from  $v$  the  $\ell$ -ball around  $v$ . Let  $\Gamma_h(v)$  denote the minimum size ball around  $v$  that contains at least  $h$  vertices. Since the maximum degree of the graph is bounded by  $d$ , it holds that  $h \leq |\Gamma_h(v)| \leq (h - 1)d$ .

The total order over the vertices induces a total order (ranking)  $\rho$  over the edges of the graph in the following straightforward manner: for any  $\{u, v\}, \{u', v'\} \in E$ ,  $\rho(\{u, v\}) < \rho(\{u', v'\})$  if and only if  $\min\{u, v\} < \min\{u', v'\}$  or  $\min\{u, v\} = \min\{u', v'\}$  and  $\max\{u, v\} < \max\{u', v'\}$ . The total order over the vertices also induces an order over those vertices visited by a Breadth First Search (BFS) starting from any given vertex  $v$ , and whenever we refer to a BFS, we mean that it is performed according to this order. Instead of writing  $\log_2(\cdot)$ , we use  $\log(\cdot)$ . We use  $\tilde{O}$  for  $\tilde{O}(x) = O(x) \cdot \text{poly}(\log x)$ .

## 2.1 Mixing-Time of Regular Graphs

Let  $G = (V, E)$  be an undirected and  $d$ -regular graph (where self-loops and multiple edges are allowed). The *Adjacency Matrix of  $G$* , denoted by  $A(G)$  is real and symmetric and so it has  $n$  real eigenvalues  $\lambda_1 \geq \lambda_2 \dots \geq \lambda_n$  where  $\lambda_1 = d$  and  $\lambda_1 > \lambda_2$  iff  $G$  is connected. A  $d$ -regular graph on  $n$  vertices is called an  $(n, d, \alpha)$ -graph if  $|\lambda_2|, |\lambda_n| \leq \alpha d$ . Define  $\lambda(G) = \max(|\lambda_2|, |\lambda_n|)$  and  $\alpha(G) = \lambda(G)/d$ . Thus, every  $d$ -regular graph on  $n$  vertices is an  $(n, d, \alpha(G))$ -graph. Let  $\hat{A}(G) = \frac{1}{d}A(G)$  denote the *normalized adjacency matrix of  $G$* . Note that  $\hat{A}^t \vec{p}$  is the distribution on the vertices resulting from random walks of length  $t$  w.r.t. the initial vertex distribution  $\vec{p}$ . We shall use the following (folklore; see, e.g., [11, 10]) theorem and lemmas.

► **Theorem 7.** *Let  $G$  be an  $(n, d, \alpha)$ -graph with the normalized adjacency matrix  $\hat{A}$ . Then for any distribution vector  $\vec{p}$  and any positive integer  $t$ :  $\|\hat{A}^t \vec{p} - \vec{u}\|_1 \leq \sqrt{n} \cdot \alpha^t$ , where  $\vec{u}$  denotes the uniform distribution vector, i.e.,  $\vec{u} \stackrel{\text{def}}{=} \frac{1}{n} \cdot \vec{1}$ .*

One can obtain from Theorem 7 an upper bound,  $t$ , such that random walks of length at least  $t$  (independent of the initial vertex distribution) yield a distribution on the vertices that is almost uniform. We denote this upper bound by  $\tau(G)$  and refer to it as the *mixing time of  $G$* . When the graph  $G$  is evident from the context, we denote the mixing time by  $\tau$ .

► **Corollary 8.** *Let  $G$  be an  $(n, d, \alpha)$ -graph with the normalized adjacency matrix  $\hat{A}$ . For any  $\tau \geq \frac{\log(2n^{3/2})}{\log(1/\alpha)}$  and for any distribution vector  $\vec{p}$  it holds that  $(\hat{A}^\tau \vec{p})_i \in [\frac{1}{2n}, \frac{3}{2n}]$ , for every  $i \in [n]$ .*

## 2.2 Mixing-Time of General Bounded Degree Graphs

Given a  $d$ -bounded degree graph  $G = (V, E)$ , we would like to relate its mixing time to  $\phi(G)$ . For this purpose, we define  $(G)_{\text{reg}}^{2d} = (V, R)$  where  $R$  contains all the edges in  $E$  and self-loops. In particular we add  $2d - d_G(v)$  self-loops to each vertex  $v \in V$ . Thus,  $(G)_{\text{reg}}^{2d}$  is a  $2d$ -regular graph. When  $d$  is clear from the context, we use  $(G)_{\text{reg}}$  to denote  $(G)_{\text{reg}}^{2d}$ .

We shall use the following classical results to relate  $\phi(G)$  and the mixing-time of  $(G)_{\text{reg}}$ .

► **Theorem 9** (Perron-Frobenius, Symmetric Case (see e.g., [30])). *Let  $G$  be a connected (weighted) graph. Let  $A(G)$  be its adjacency matrix, and let  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$  be its eigenvalues. Then,*

- (1)  $\lambda_1$  has a strictly positive eigenvector,
- (2)  $\lambda_1 \geq -\lambda_n$ , and
- (3)  $\lambda_1 > \lambda_2$ .

► **Lemma 10** (Cheeger's Inequality [5]). *Let  $G$  be a  $d$ -regular graph with eigenvalues  $\lambda_1 \geq \lambda_2 \dots \geq \lambda_n$ . Then,  $\frac{d-\lambda_2}{2d} \leq \phi(G) \leq \frac{\sqrt{2d(d-\lambda_2)}}{d}$ .*

We first prove (following [29, P. 106]) that the maximum eigenvalue in absolute value of  $A((G)_{\text{reg}})$  is  $\lambda_2$ .

▷ Claim 11. Let  $G$  be a connected,  $d$ -bounded degree graph. Let  $\{\lambda_i\}_i$  denote the eigenvalues of  $A((G)_{\text{reg}})$ , then  $\max(|\lambda_2|, |\lambda_n|) = \lambda_2$ .

The following claim, combined with Corollary 8 provides an upper bound on the mixing-time of  $(G)_{\text{reg}}$ .

▷ Claim 12. Let  $G = (V, E)$  be a connected  $d$ -bounded degree graph on  $n$  vertices, then  $(G)_{\text{reg}}$  is an  $\left(n, 2d, \left(1 - \frac{\phi^2(G)}{2}\right)\right)$ -graph.

The next Corollary follows from Corollary 8 and Claim 12.

► **Corollary 13.** *Let  $G = (V, E)$  be a connected  $d$ -bounded degree graph on  $n$  vertices and let  $v \in V$ . If we perform a random-walk in  $(G)_{\text{reg}}$ , starting from  $v$ , of length at least  $\tau(G) \stackrel{\text{def}}{=} \frac{\log(2n^{3/2})}{\log\left(\left(1 - \frac{\phi^2(G)}{2}\right)^{-1}\right)}$ , then the probability this walk ends in  $u$  is at least  $1/(2n)$  for every  $u \in V$ .*

### 3 LSSG for a Single Cluster

In this section, we prove Theorem 4, in particular, we present and prove correct our LSSG algorithm for the case where the input graph is a single cluster, i.e., a  $(1, \phi)$ -clusterable graph. As such, our algorithm receives  $\phi$  as a parameter where  $\phi$  is a lower bound on the conductance of the input graph. We first describe our algorithm from a global point of view (Subsection 3.1) and then explain how this global algorithm can be implemented locally (Subsection 3.3).

#### 3.1 The Global Algorithm for a Single Cluster

The algorithm has two stages. In the first stage, it picks an arbitrary vertex,  $\mathcal{P}$ , as the *primary-center*. It then performs  $r \stackrel{\text{def}}{=} \tilde{\Theta}(\sqrt{n})$   $s$ -wise independent lazy random walks from  $\mathcal{P}$  and takes the end-vertices of these walks to be the set of *secondary centers* (Steps 4-5). As defined in Section 2, when performing a lazy random walk, in each step, an edge is selected uniformly and independently with probability  $1/2d$  (recall that we add  $2d - d_G(v)$  self-loops to each vertex  $v$  to obtain a  $2d$ -regular graph). The edges traversed by the random walks (which are not self-loops) are added to the spanner (Step 6). This guarantees that all the secondary centers are connected in the spanner. The length of the random walks  $\tau$  is determined by Corollary 8 to ensure that the secondary centers are distributed almost uniformly in  $V$ .

In the second stage, the algorithm constructs spanning trees of the Voronoi cells, which are defined w.r.t. the secondary centers (Steps 7-8). Since the secondary centers are distributed almost uniformly in  $V$ , with high-probability, all vertices  $v \in V$  will have a secondary center in  $\Gamma_h(v)$  where  $h \stackrel{\text{def}}{=} \sqrt{n}$ . If this is not the case (i.e., no secondary center is found in  $\Gamma_h(v)$ ), then all the edges incident to  $v$  are added to the spanner (Step 9). Note that removing Step 9 yields an LSSG algorithm that spans  $G$  w.h.p., while including it yields an algorithm that outputs a spanning subgraph with probability 1.

■ **Algorithm 1** Globally Computing a Sparse Spanning Subgraph.

---

**Input:** A graph  $G = (V, E)$  with conductance at least  $\phi$  and maximum degree bounded by  $d$ .

**Output:**  $G' = (V, E')$  is a sparse spanning subgraph of  $G$  w.h.p.

- 1  $E' \leftarrow \emptyset$ .
- 2 Select a *primary-center*  $\mathcal{P} \in V$  arbitrarily.
- 3 Let  $r \stackrel{\text{def}}{=} \Theta(\sqrt{n} \cdot \log n)$ ,  $h \stackrel{\text{def}}{=} \sqrt{n}$ ,  $s \stackrel{\text{def}}{=} \Theta(\log n)$ ,  $\tau \stackrel{\text{def}}{=} \Theta\left(\frac{\log n}{\phi^2}\right)$ .
- 4 Perform  $r$   $s$ -wise independent lazy random walks  $\mathcal{R} \stackrel{\text{def}}{=} \{\rho_1, \dots, \rho_r\}$  where  $\rho_i = (v_1^{(i)} = \mathcal{P}, \dots, v_\tau^{(i)})$ .
- 5 Let  $\mathcal{S}$  denote the set of *secondary centers* defined as follows  $\mathcal{S} \stackrel{\text{def}}{=} \bigcup_{i \in [r]} \{v_\tau^{(i)}\}$ . // The end-vertices of the random-walks are selected to be the set of secondary centers.
- 6  $E' \leftarrow E(\mathcal{R}) \cap E$ . // where  $E(\mathcal{R})$  denotes the edge set of the random walks in  $\mathcal{R}$ . Add all the edges traversed by the random walks (that are not self-loops) in  $\mathcal{R}$  to the set  $E'$ .
- 7 Every vertex  $v \in V$  assigns itself to the closest secondary center in  $\Gamma_h(v)$  denoted by  $\sigma(v)$  (break ties by choosing the one with the smallest ID). If  $\Gamma_h(v) \cap \mathcal{S} = \emptyset$  then set  $\sigma(v) = \perp$ .
- 8 For each  $s \in \mathcal{S}$ , let  $\text{Vor}(s) \stackrel{\text{def}}{=} \{v \in V \mid \sigma(v) = s\}$ . Let  $\text{STree}(s)$  denote the BFS tree that spans  $\text{Vor}(s)$ .  $E' \leftarrow E' \cup \text{STree}(s)$ .
- 9 Add to  $E'$  all the edges incident to vertices,  $v$ , such that  $\sigma(v) = \perp$ .
- 10 **return**  $G' = (V, E')$ .

---

### 3.2 Correctness of the Global Algorithm for a Single Cluster

To prove the correctness of the global algorithm, we claim that w.h.p. when the algorithm stops, there is a path in  $G'$  from each vertex to a secondary center and that this path is short; We begin by proving that w.h.p., every vertex  $v \in V$  has a secondary center in  $\Gamma_h(v)$ . We shall use the following concentration bound, which applies for  $s$ -wise independent random variables defined as follows.<sup>7</sup>

► **Definition 14** ( $s$ -wise independence). *A set of discrete random variables  $X_1, \dots, X_n$  are called  $s$ -wise independent if for any set  $I \subseteq \{1, \dots, n\}$  with  $|I| \leq s$  and any values  $x_i$  we have  $\Pr[\bigwedge_{i \in I} (X_i = x_i)] = \prod_{i \in I} \Pr[X_i = x_i]$ .*

Note that the random walks that the algorithm performs are also random variables. In particular, this is a set of  $s$ -wise independent random walks, i.e., any subset of size at most  $s$  of walks (out of the  $r$  walks that the algorithm performs) are mutually independent.

► **Theorem 15** (Theorem 5(III) in [28]). *If  $X$  is a sum of  $s$ -wise independent random variables, each of which is in the interval  $[0, 1]$  with  $\mu = \mathbb{E}(X)$ , then For  $\delta \leq 1$  and  $s \leq \lfloor \delta^2 \mu e^{-1/3} \rfloor$ , it holds that  $\Pr[|X - \mu| \geq \delta \mu] \leq e^{-\lfloor s/2 \rfloor}$ .*

▷ **Claim 16.** With high probability, for every  $u \in V$ ,  $\Gamma_h(u) \cap \mathcal{S} \neq \emptyset$ .

*Proof.* Fix  $\mathcal{P}$  and let  $v$  be a vertex in  $V$ . Consider a random walk on  $(G)_{\text{reg}}$  (or alternatively, a lazy random-walk on  $G$ ) that starts at  $\mathcal{P}$ . By Corollary 13, when performing a random-walk from  $\mathcal{P}$  of length  $\tau$  in  $(G)_{\text{reg}}$ , the probability that  $v$  is the end-vertex of the random walk is at

---

<sup>7</sup> Having bounded independence reduces the number of random bits used by the local implementation of this algorithm, as shown in the next section.

least  $\frac{1}{2n}$ . Let  $\mathcal{E}_i^v$  denote the event that the  $i$ -th random walk ended at  $v$ , namely the event that  $v_r^{(i)} = v$ , and let  $X_i^v$  denote the indicator variable for this event. Thus,  $\Pr[\mathcal{E}_i^v] = \mathbb{E}[X_i^v] \geq \frac{1}{2n}$ . Let  $H \subseteq V$  be a set of cardinality  $h$  and let  $\mathcal{E}_i^H$  denote the event that the  $i$ -th random walk ended at  $H$ . Let  $X_i^H$  denote the indicator variable for this event. Since the events  $\{\mathcal{E}_i^v\}_{v \in V}$  are disjoint (i.e. mutually exclusive) we obtain that  $\mathbb{E}[X_i^H] = \Pr[\mathcal{E}_i^H] = \sum_{v \in H} \Pr[\mathcal{E}_i^v] \geq \frac{h}{2n}$ . By linearity of expectation,  $\mathbb{E}\left[\sum_{i \in [r]} X_i^H\right] \geq \frac{hr}{2n} = \Theta(\log n)$ , where the last equality follows since  $h = \sqrt{n}$ . Since we perform  $r$   $s$ -wise independent random walks from  $\mathcal{P}$ , the random variables  $\{X_i^H\}_{i \in [r]}$  are also  $s$ -wise independent. Set  $Y \stackrel{\text{def}}{=} \sum_{i \in [r]} X_i^H$  and  $\mu \stackrel{\text{def}}{=} \mathbb{E}[Y] = \Theta(\log n)$ . By Theorem 15,  $\Pr[|Y - \mu| \geq \mu/2] \leq e^{-\lfloor s/2 \rfloor}$  where  $s \stackrel{\text{def}}{=} \lfloor \frac{1}{4} \left(\frac{hr}{2n}\right) e^{-1/3} \rfloor \leq \lfloor \frac{1}{4} \mu e^{-1/3} \rfloor$ . If  $|Y - \mu| < \mu/2$ , then  $Y > \mu/2 \geq 1$ . Thus, the probability that none of the  $r$  random-walks ends in  $H$  is at most  $e^{-\lfloor s/2 \rfloor} = 1/n^c$ , where  $c$  is determined by the exact setting of  $r$ . Hence, by the union bound over all vertices, we obtain that with high probability  $\Gamma_h(v) \cap \mathcal{S} \neq \emptyset$  for every  $v \in V$ .  $\triangleleft$

The following claim (the proof of which appears in Appendix C) argues that BFS trees are of height logarithmic in the number of their vertices.

$\triangleright$  **Claim 17.** Let  $G = (V, E)$  be a  $d$ -bounded degree graph and let  $\phi$  be a lower bound on  $\phi(G)$ . For any  $v \in V$  and  $x \leq |V|/2$ , the  $\ell$ -ball centered at  $v$  contains at least  $x$  vertices provided that  $\ell \geq \frac{\log x}{\log(1+\phi)}$ .

Claim 17 implies that the paths' length from every vertex to its secondary center is logarithmic in  $n$ , formalized as follows.

$\blacktriangleright$  **Corollary 18.** Let  $s \in \mathcal{S}$ . Then the depth of  $\text{STree}(s)$  is at most  $\frac{\log h}{\log(1+\phi)}$ .

We now prove the main theorem of this section, which bounds the stretch factor and size of the graph,  $G' = (V, E')$  obtained by Algorithm 1. In particular, for constant  $\phi$  it follows that  $|E'| = n + o(n)$ , and the stretch is  $\Theta(\log n)$ .

$\blacktriangleright$  **Theorem 19.** Algorithm 1 computes a Sparse Spanning Graph of  $G$ ,  $G' = (V, E')$ , such that:

- (1) The attained stretch is  $\Theta\left(\frac{\log n}{\phi^2}\right)$ , and
- (2)  $|E'| = n + O\left(\frac{\sqrt{n} \log^2 n}{\phi^2}\right)$  with high probability.

**Proof.** We first prove Item 1 of the claim. Consider an edge  $\{u, v\}$  which belongs to  $E$  but not to  $E'$ . We first note that it follows that both  $\sigma(u) \neq \perp$  and  $\sigma(v) \neq \perp$  because otherwise  $\{u, v\}$  is added to  $E'$  in Step 9. If  $u$  and  $v$  belong to the same Voronoi cell then the distance between them in  $G'$  is at most  $2 \frac{\log \sqrt{n}}{\log(1+\phi)}$  (since, by Corollary 18, the distance from every vertex to its secondary center is at most  $\frac{\log \sqrt{n}}{\log(1+\phi)}$ ). If  $u$  and  $v$  do not belong to the same Voronoi cell, then the distance in  $G'$  between their respective centers,  $\sigma(u)$  and  $\sigma(v)$  is at most  $2\tau$  (because the distance in  $G'$  between every secondary center to  $\mathcal{P}$  is at most  $\tau$ ), where  $\tau \stackrel{\text{def}}{=} \frac{\log(2n^{3/2})}{\log\left(\left(1 - \frac{\phi^2}{2}\right)^{-1}\right)}$ . Thus, in this case the distance in  $G'$  between  $u$  and  $v$  is at most  $2 \cdot \left(\tau + \frac{\log \sqrt{n}}{\log(1+\phi)}\right)$ . Since for all  $x \geq 0$  it holds that  $x - x^2/2 \leq \ln(1+x)$ , it follows that the attained stretch is  $\Theta\left(\frac{\log n}{\phi^2}\right)$ , as claimed.

Item 2 of the claim follows from the construction. More specifically, at Step 6, we add at most  $r \cdot \tau$  edges to  $E'$ . In Step 8, we add at most  $n - 1$  edges since the Voronoi cells are vertex-disjoint. Finally, by Claim 16, with high probability, in Step 9, we do not add any edges to  $E'$ .  $\blacktriangleleft$



### 3.3 Details of the Implementation of the Local Algorithm for a Single Cluster

In this section, we describe how to implement Algorithm 1 locally. We also bound the query and time complexity of the local algorithm as well as the total number of bits it uses. We conclude this section with the proof of Theorem 4.

**Performing lazy random-walks in  $G$ .** Performing a lazy-random walk of length  $\ell$  in  $G$  requires at most  $\ell$  probes to  $G$ . In each step, we select an index uniformly at random from  $i \in [2d]$  and perform a neighbor-query  $(v, i)$  to reveal the  $i$ th neighbor of  $v$ , where  $v$  denotes the ID of the current vertex.<sup>8</sup> If the probe returns a vertex ID, then we move to that neighbor; otherwise, the walk stays at  $v$ . We note that performing a lazy-random walk in  $G$  is equivalent to performing a random walk in  $(G)_{\text{reg}}$ .

**Bounding the Number of Random Bits.** Following [23], we shall use the classical result from [31] for obtaining random bits with bounded independence in a local manner.

► **Definition 20.** For  $N, M, s \in \mathbb{N}$  such that  $s \leq N$ , a family of functions  $\mathcal{H} = \{h : [N] \rightarrow [M]\}$  is  $s$ -wise independent if for all distinct  $x_1, \dots, x_s \in [N]$ , the random variables  $h(x_1), \dots, h(x_s)$  are independent and uniformly distributed in  $[M]$  when  $h$  is chosen randomly from  $\mathcal{H}$ .

► **Lemma 21** (Corollary 3.34 in [31]). For every  $\gamma, \beta, s \in \mathbb{N}$ , there is a family of  $s$ -wise independent functions  $\mathcal{H}_{\gamma, \delta} = \{h : \{0, 1\}^\gamma \rightarrow \{0, 1\}^\delta\}$  such that choosing a random function from  $\mathcal{H}_{\gamma, \delta}$  takes  $s \cdot \max\{\gamma, \delta\}$  random bits, and evaluating a function from  $\mathcal{H}_{\gamma, \delta}$  takes time  $\text{poly}(\gamma, \delta, s)$ .

▷ **Claim 22.** Performing  $r$   $s$ -wise independent random walks from  $\mathcal{P}$  in  $(G)_{\text{reg}}$  can be implemented by using  $O(\log n(\log n + \tau \log d))$  random bits and time-complexity  $r \cdot \text{poly}(\tau, \log n)$ .

Proof. Performing a single random-walk in  $(G)_{\text{reg}}$  of length  $\tau$  from  $\mathcal{P}$  requires  $O(\tau \log d)$  random bits (since we are performing  $\tau$  steps and in each step, we are selecting an edge u.a.r. out of  $2d$  edges). Let  $\gamma \stackrel{\text{def}}{=} \Theta(\log r)$  denote the number of bits that are required to indicate the index of the random walk in  $[r]$  and let  $\delta \stackrel{\text{def}}{=} \Theta(\tau \log d)$  denote the number of random bits that are required for performing a single walk. To perform  $r$   $s$ -wise independent random walks, it suffices to choose a random function from a family of  $s$ -wise independent functions,  $\mathcal{H}_{\gamma, \delta}$  which takes as parameter the index of the random walk and returns  $\delta$  bits. By Lemma 21 this can be done by using  $s \cdot \max\{\gamma, \delta\} = O(\log n(\log n + \tau \log d))$  random bits. Furthermore, the time complexity of retrieving the bits for performing a single random walk is  $\text{poly}(\gamma, \delta, s) = \text{poly}(\tau, \log n)$ . The claim follows. ◁

**Locally Computing a Sparse Spanning Subgraph.** On query  $\{u, v\}$ , the local algorithm first performs the random walks as listed in Algorithm 1, Step 4. If  $\{u, v\} \in E(\mathcal{R})$ , then the algorithm returns YES. Otherwise, it finds  $\sigma(u)$  and  $\sigma(v)$ . If either  $\sigma(u) = \perp$  or  $\sigma(v) = \perp$  then it returns YES. Otherwise, if  $\sigma(u) = \sigma(v)$ , then the algorithm returns YES iff  $\{u, v\} \in \text{STree}(\sigma(v))$ . Otherwise, if  $\sigma(u) \neq \sigma(v)$ , the algorithm returns NO. Finding  $\sigma(v)$

<sup>8</sup> A common assumption in LCA in general and hence also in LSSG's is that the algorithm knows  $n$ . Similarly, as in [17], for bounded-degree graphs, the bound on the maximum vertex degree  $d$  of the graph instance at hand is also known to the LSSG algorithm.

can be done by making  $O(hd^2)$  queries. To see this, observe that the number of vertices we explore by performing a BFS, layer by layer, until we first see at least  $h$  vertices is at most  $(h-1)d$ . The subgraph induced on these vertices contains at most  $hd^2$  edges, thus the query and time complexity of this step is indeed  $O(hd^2)$ <sup>9</sup>. Checking if  $\{u, v\} \in \text{STree}(\sigma(v))$  can be implemented at the same cost. To see this, observe that when we find  $\sigma(v)$  as described above, then we also reveal all the shortest paths between  $v$  and  $\sigma(v)$  in  $G$ . Since we can decide which one of these paths is the path between  $v$  and  $\sigma(v)$  in  $\text{STree}(\sigma(v))$ , we can decide if  $\{u, v\}$  belongs to this path at the same cost of finding  $\sigma(v)$  and  $\sigma(u)$ <sup>10</sup>. This concludes the description of the local implementation of Algorithm 1. We are now ready to prove Theorem 4.

**Proof of Theorem 4.** The correctness of the algorithm, that is, Item (1), follows from Theorem 19. As described above (in the analysis of the local implementation), the algorithm makes  $r \cdot \tau + hd^2 = O\left(\sqrt{n} \cdot \left(\frac{\log^2 n}{\phi^2} + d^2\right)\right)$  graph queries. By Claim 22 the local implementation of Algorithm 1 uses  $O(\log n(\log n + \tau \log d)) = O\left(\frac{\log d \cdot \log^2 n}{\phi^2}\right)$  random bits, which concludes the proof of the theorem. ◀

## 4 LSSG for Clusterable Graphs

In this section, we prove Theorem 5. First, we describe our LSSG algorithm that works under the promise that the input graph is a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph where each cluster is of size at least  $\beta n$ .

### 4.1 The Global Algorithm for Clusterable graphs

The listing of our global algorithm appears as Algorithm 2. We next describe how it proceeds, step by step. All the parameters we will mention are defined in Step 2 of the algorithm. Initially, the set of edges of the spanner,  $E'$ , is empty (Step 1). The algorithm begins with selecting  $p$  primary-centers uniformly at random from  $V$  (Step 3). It then performs  $r$   $s$ -wise independent lazy-random walks from each primary center (Step 4). It picks the endpoints of these random-walks to be the set of *secondary-centers* (Step 5). Additionally, all the edges traversed by these random walks are added to  $E'$  (Step 4). After that, the vertices of the graph are partitioned as follows. Each one of the vertices,  $v$ , joins the cell of the secondary center, which is closest to  $v$  in  $\Gamma_h(v)$ , breaking ties by ID (Step 6). If  $\Gamma_h(v)$  does not include a secondary center, then all the edges that are incident to  $v$  are added to  $E'$  (Step 9). In Step 10 the algorithm picks  $t$  random pairs from  $V \times [d]$ . For each such pair,  $(v, i)$ , such that  $v$  has an  $i$ -th neighbor, the edge  $\{u, v\}$  is added to  $E'$  where  $u$  is the  $i$ -th neighbor of  $v$  (Step 10). We define an artificial-cluster to be a maximal set of vertices that agree on their primary-center (see formal definition in Step 7). For every pair of artificial-clusters such that  $E'$  does not include an edge from their cut, all the edges in the corresponding cut are added to  $E'$  (Step 11). This concludes the description of the algorithm.

### 4.2 Correctness of the Global Algorithm for Clusterable Graphs

To prove the correctness of the algorithm, we need to show that the obtained subgraph spans the graph with a logarithmic stretch and that it is sparse. Proving that the obtained subgraph spans the graph is relatively straightforward and follows by the design of the

<sup>9</sup> This procedure appears in [17] as Algorithm **Find-Center**.

<sup>10</sup> See more details in algorithm **Get BFS outgoing-edges endpoints** in [17].

■ **Algorithm 2** Globally Computing a Sparse Spanning Subgraph of a Clusterable Graph.

- 
- Input:** A connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph  $G = (V, E)$  where each cluster is of size at least  $\beta|V|$ .
- Output:**  $G' = (V, E')$  is a sparse spanning subgraph of  $G$  w.h.p.
- 1  $E' \leftarrow \emptyset$ .
  - 2 Let  $p \stackrel{\text{def}}{=} \Theta(\beta^{-1} \log n)$ ,  $r \stackrel{\text{def}}{=} \Theta(\frac{\log n}{\tau \phi_{\text{out}}})$ ,  $s \stackrel{\text{def}}{=} \Theta(\log n)$ ,  $\tau \stackrel{\text{def}}{=} \Theta(\frac{\log n}{\phi_{\text{in}}^2})$ , and  $h \stackrel{\text{def}}{=} \Theta(k\tau\phi_{\text{out}}n)$ .
  - 3 Select  $p$  primary-centers  $\mathcal{P} \stackrel{\text{def}}{=} \{\psi_1, \dots, \psi_p\}$  u.a.r. in  $V$ .
  - 4 From each primary-center,  $\psi \in \mathcal{P}$ , perform  $r$   $s$ -wise independent lazy random-walks. Add all the edges in  $E$  traversed by these random walks to  $E'$ .
  - 5 Let  $\mathcal{S}$  denote the set of endpoints of these random-walks. We refer to  $\mathcal{S}$  as the set of secondary centers. We say that the primary-center of  $v \in \mathcal{S}$  is  $\psi$  if the random-walk that ended in  $v$  started at  $\psi$  (break ties by choosing the one with the smallest ID).
  - 6 Every vertex  $v \in V$  assigns itself to the closest secondary center in  $\Gamma_h(v)$  denoted by  $\sigma(v)$  (break ties by choosing the one with the smallest ID). If  $\Gamma_h(v) \cap \mathcal{S} = \emptyset$  then set  $\sigma(v) = \perp$ .
  - 7 The primary-center of  $v$  is set to be the primary-center of  $\sigma(v)$ . The *artificial-cluster* of  $v$  is defined to be the set of all vertices whose primary-center equals the primary-center of  $v$ .
  - 8 For each  $s \in \mathcal{S}$ , let  $\text{Vor}(s) \stackrel{\text{def}}{=} \{v \in V \mid \sigma(v) = s\}$ . Let  $\text{STree}(s)$  be a spanning tree of  $\text{Vor}(s)$ .  $E' \leftarrow E' \cup \text{STree}(s)$ .
  - 9 Add to  $E'$  all the edges that are incident to vertices,  $v$ , such that  $\sigma(v) = \perp$ .
  - 10 Select  $t \stackrel{\text{def}}{=} \Theta(\varepsilon^{-1}k^2d \log n)$  pairs u.a.r. from  $V \times [d]$ . Let  $\mathcal{T}$  denote the set of edges that correspond to these pairs (an edge  $\{u, v\}$  corresponds to the pair  $(v, i)$  if  $u$  is the  $i$ -th neighbor of  $v$ ).  $E' \leftarrow E' \cup \mathcal{T}$ .
  - 11 Add to  $E'$  all the edges between artificial-clusters not connected by an edge in  $\mathcal{T}$ .
  - 12 **return**  $G' = (V, E')$ .
- 

algorithm. Both the proof of the low stretch and sparsity of the obtained spanner appears in Theorem 28, where for the sparsity proof, we need to show that the number of edges that we add in Steps 4, and 8-11 is not too much. The proof that the obtained spanner is of a low stretch is the main technical challenge of this section. To this end, it is sufficient to show that w.h.p. for every  $v \in V$ ,  $\Gamma_h(v)$  includes a secondary-center (see Claim 26). To this end, we next analyze the distribution of the endpoints of the lazy-random walks that the algorithm performs.

For any subset  $C \subseteq V$ , we slightly abuse notation and denote by  $(C)_{\text{reg}}$  the  $2d$ -regular graph  $(G[C])_{\text{reg}}$ , where  $G[C]$  is the subgraph induced on  $C$  in  $G$ .

Throughout this section,  $G = (V, E)$  is a  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph,  $C \subseteq V$  is a  $(\phi_{\text{in}}, \phi_{\text{out}})$ -cluster of  $G$  and  $S$  is the set of self-loops that are added to  $(C)_{\text{reg}}$  due to the edges in the cut between  $C$  and  $V \setminus C$ . Namely, for each  $v \in C$ ,  $S$  contains  $e(\{v\}, V \setminus C)$  self-loops of  $v$ .

For every  $u, v \in C$ , let  $p^{\ell, v}(u)$  denote the probability that an  $\ell$ -length random-walk that starts at  $v$  in  $(C)_{\text{reg}}$  ends at  $u$ . Thus  $\sum_{u \in C} p^{\ell, v}(u) = 1$ . We let  $p_{\text{bad}}^{\ell, v}(u)$  denote the probability that an  $\ell$ -length random-walk in  $(C)_{\text{reg}}$  that starts at  $v$  ends in  $u$  and traverses an edge of  $S$ . Let  $p_{\text{good}}^{\ell, v}(u) \stackrel{\text{def}}{=} p^{\ell, v}(u) - p_{\text{bad}}^{\ell, v}(u)$ .

▷ **Claim 23.** For at least half of the vertices  $v \in C$ , it holds that  $\sum_{u \in C} p_{\text{bad}}^{\ell, v}(u) \leq \ell\phi_{\text{out}}$ . We call such a vertex,  $v$ , useful.

*Proof.* We prove that if we perform an  $\ell$ -length random-walk,  $\rho$ , in  $(C)_{\text{reg}}$  from a vertex  $v$  selected u.a.r. from  $C$ , then the probability that  $\rho$  traverses an edge from  $S$  is at most  $\ell\phi_{\text{out}}/2$ . In other words, we will show that  $\mathbb{E}[\sum_{u \in C} p_{\text{bad}}^{\ell, v}(u)] \leq \ell\phi_{\text{out}}/2$ , where the expectation is taken over  $v$  which is selected u.a.r. from  $C$ . The claim will then follow by Markov's inequality.

Since  $(C)_{\text{reg}}$  is  $2d$ -regular and  $v$  is selected u.a.r. from  $C$ , all the vertices in  $\rho$  are distributed uniformly at random from  $C$  as well (since  $P\vec{u} = \vec{u}$  for every doubly-stochastic matrix  $P$ ). The probability of traversing an edge from  $S$  when we select a vertex u.a.r. from  $C$  and then take a single step from this vertex is  $\frac{|S|}{2d|C|}$ . This is simply because each one of the edges in  $S$  is traversed with probability  $\frac{1}{|C|} \cdot \frac{1}{2d}$  (recall that the edges in  $S$  correspond to self-loops). Thus, by union bound, the probability of traversing an edge from  $S$  in one of the  $\ell$  steps of the random walk is at most  $\frac{\ell|S|}{2d|C|}$ . Since  $\frac{|S|}{d|C|} \leq \phi_{\text{out}}$  we obtain that this probability is at most  $\ell\phi_{\text{out}}/2$ , as desired.  $\triangleleft$

The following claim relates random-walks in  $(G)_{\text{reg}}$  to random-walks in  $(C)_{\text{reg}}$ .

$\triangleright$  **Claim 24.** If we perform an  $\ell$ -length random-walk in  $(G)_{\text{reg}}$  from  $v \in C$  then the probability that this random walk ends at  $u \in C$  is at least  $p_{\text{good}}^{\ell,v}(u)$ .

*Proof.* Recall that  $p_{\text{good}}^{\ell,v}(u)$  is the probability that an  $\ell$ -length random-walk in  $(C)_{\text{reg}}$  that starts at  $v$  ends in  $u$  and does not traverse an edge of  $S$ . The edge set of  $(C)_{\text{reg}}$  includes parallel self-loops. If we identify each edge by its endpoints and the labels of its ports, then we get that from each vertex  $v \in C$ , there are exactly  $(2d)^\ell$  distinct paths of length  $\ell$  in  $(C)_{\text{reg}}$ . Let  $P^v$  denote the set of these paths. When we perform a random walk in  $(C)_{\text{reg}}$  from  $v$ , each path in  $P^v$  occurs with probability  $1/(2d)^\ell$ . Moreover, for every  $u \in C$ , let  $P_u^v$  denote the set of paths in  $P^v$  that end at  $u$  and do not traverse an edge from  $S$ . Each path in  $P_u^v$  contributes  $1/(2d)^\ell$  to  $p_{\text{good}}^{\ell,v}(u)$  and each path in  $P^v \setminus P_u^v$  contributes 0. Assume, w.l.o.g., that  $(C)_{\text{reg}}$  is obtained from  $G$  by first converting  $G$  to  $(G)_{\text{reg}}$  and then replacing the edges in the cut  $E(C, V \setminus C)$  with self-loops. This way, every self-loop in  $(C)_{\text{reg}}$  which does not belong to  $S$  also appears in  $(G)_{\text{reg}}$  (when taking into account the port numbers). Thus, if we perform a random-walk in  $(G)_{\text{reg}}$  from  $v$ , each path from  $P_u^v$  occurs with probability  $1/(2d)^\ell$  as well. The claim follows.  $\triangleleft$

$\triangleright$  **Claim 25.** Let  $v$  be a vertex which is useful w.r.t.  $C$ . For all  $u \in C$ , except for at most  $4\tau\phi_{\text{out}}|C|$  vertices, it holds that a  $\tau$ -length random-walk in  $(G)_{\text{reg}}$  from  $v$  ends at  $u$  with probability at least  $(4n)^{-1}$ .

*Proof.* Let  $v$  be a vertex, which is useful w.r.t.  $C$ , and let  $\ell \in \mathbb{N}$ . By Claim 23,  $\sum_{u \in C} p_{\text{bad}}^{\ell,v}(u) \leq \ell\phi_{\text{out}}$ . Therefore  $\mathbb{E}[p_{\text{bad}}^{\ell,v}(u)] \leq \frac{\ell\phi_{\text{out}}}{|C|}$ , where the expectation is taken over  $u$  selected u.a.r. from  $C$ . Thus, by Markov's inequality for at most  $\gamma$ -fraction of the vertices  $u \in C$  it holds that  $p_{\text{bad}}^{\ell,v}(u) > \frac{\ell\phi_{\text{out}}}{\gamma|C|}$ .

By replacing  $\ell$  with  $\tau$  and  $\gamma$  with  $4\tau\phi_{\text{out}}$  we obtain that for at least  $(1 - 4\tau\phi_{\text{out}})$ -fraction of the vertices  $u \in C$  it holds that  $p_{\text{good}}^{\tau,v}(u) \geq p^{\tau,v}(u) - \frac{1}{4|C|}$ . Since  $C$  is a  $(\phi_{\text{in}}, \phi_{\text{out}})$ -cluster in  $G$  it follows from Corollary 13 that  $p^{\tau,v}(u) \geq 1/(2|C|)$  for every  $u$ . Thus it holds that  $p_{\text{good}}^{\tau,v}(u) \geq 1/(4|C|) \geq 1/(4n)$ . Hence, the claim follows from Claim 24.  $\triangleleft$

$\triangleright$  **Claim 26.** Let  $G = (V, E)$  be a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph and let  $C_1, \dots, C_f$  be a partition of  $G$  into  $f \leq k$   $(\phi_{\text{in}}, \phi_{\text{out}})$ -clusters. If for every  $i \in [f]$ ,  $C_i \cap \mathcal{P}$  contains a useful vertex w.r.t.  $C_i$ , then w.h.p. for every  $v \in V$ ,  $\Gamma_h(v) \cap \mathcal{S} \neq \emptyset$ .

*Proof.* Let  $G = (V, E)$  be a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph and let  $C_1, \dots, C_f$  be a partition of  $G$  into  $f \leq k$   $(\phi_{\text{in}}, \phi_{\text{out}})$ -clusters. Assume for every  $i \in [f]$ ,  $C_i \cap \mathcal{P}$  contains a useful vertex,  $u_i$ , w.r.t.  $C_i$ . Let  $H$  be a set of vertices of cardinality at least  $h$  and let  $\gamma = 4\tau\phi_{\text{out}}$  where  $h$  and  $\tau$  are as defined in Step 2 of Algorithm 2. By Claim 25, for every  $i \in [f]$  and for all vertices  $u \in C_i$  except for at most  $\gamma|C_i|$  it holds that an  $\tau$ -length random-walk in  $(G)_{\text{reg}}$  from  $u_i$  ends at  $u$  with probability at least  $(4n)^{-1}$ . Let us call these vertices *good* w.r.t.  $C_i$ .

Let  $C_j$  be the dominant cluster in  $H$ . Namely, the cluster which maximizes the intersection with  $H$ . By an averaging argument  $C_j \cap H \geq h/k \geq 5\tau\phi_{\text{out}}n$ , where the last inequality holds for an appropriate setting of  $h$ . Since the number of vertices in  $C_j$  which are not good (w.r.t.  $C_j$ ) is at most  $\gamma|C_j| \leq 4\tau\phi_{\text{out}}n$ , we obtain that the number of good vertices in  $C_j$  is at least  $\tau\phi_{\text{out}}n$ . From this point, the rest of the proof is similar to the proof of Claim 16. For  $v \in V$ , let  $\mathcal{E}_i^v$  denote the event that the  $i$ -th random walk from  $u_j$  ended at  $v$  and let  $X_i^v$  denote the indicator variable for this event. Thus, for every good vertex,  $v$ , w.r.t.  $C_j$  holds that  $\Pr[\mathcal{E}_i^v] = \mathbb{E}[X_i^v] \geq \frac{1}{4n}$ . Let  $\mathcal{E}_i^H$  denote the event that the  $i$ -th random walk from  $u_j$  ended at  $H$ . Let  $X_i^H$  denote the indicator variable for this event. Since the events  $\{\mathcal{E}_i^v\}_{v \in V}$  are disjoint (i.e. mutually exclusive) we obtain that  $\mathbb{E}[X_i^H] = \Pr[\mathcal{E}_i^H] = \sum_{v \in H} \Pr[\mathcal{E}_i^v] \geq \frac{\tau\phi_{\text{out}}n}{4n} = \frac{\tau\phi_{\text{out}}}{4}$ . By linearity of expectation,  $\mathbb{E}\left[\sum_{i \in [r]} X_i^H\right] \geq r \cdot \frac{\tau\phi_{\text{out}}}{4} = \Theta(\log n)$ . Since we perform  $r$   $s$ -wise independent random walks from  $u_j$ , the random variables  $\{X_i^H\}_{i \in [r]}$  are also  $s$ -wise independent. Set  $Y \stackrel{\text{def}}{=} \sum_{i \in [r]} X_i^H$  and  $\mu \stackrel{\text{def}}{=} \mathbb{E}[Y] = \Theta(\log n)$ . By Theorem 15,  $\Pr[|Y - \mu| \geq \mu/2] \leq e^{-\lfloor s/2 \rfloor}$  where  $s \stackrel{\text{def}}{=} \lfloor \frac{1}{4} \left( \frac{r\tau\phi_{\text{out}}}{4} \right) e^{-1/3} \rfloor \leq \lfloor \frac{1}{4} \mu e^{-1/3} \rfloor$ . If  $|Y - \mu| < \mu/2$ , then  $Y > \mu/2 \geq 1$ . Thus, the probability that none of the  $r$  random-walks ends in  $H$  is at most  $e^{-\lfloor s/2 \rfloor} = 1/n^c$ , where  $c$  is determined by the exact setting of  $r$ . Thus, by union bound over all vertices, we obtain that with high probability  $\Gamma_h(v) \cap \mathcal{S} \neq \emptyset$  for every  $v \in V$ .  $\triangleleft$

We say that a pair of artificial-clusters are *heavy* if the number of edges in their edge-cut is at least  $\varepsilon n/(2k^2)$ .

$\triangleright$  **Claim 27.** With high probability,  $\mathcal{T}$  contains an edge from the cut of every pair of heavy artificial-clusters.

*Proof.* Let  $C_1$  and  $C_2$  be a heavy pair of artificial-clusters. By definition, the number of edges in their cut is at least  $\varepsilon n/(2k^2)$ . Thus the probability that a pair chosen u.a.r. from  $V \times [d]$  hits this cut is at least  $\frac{\varepsilon n}{2k^2} \cdot \frac{1}{dn} = \frac{\varepsilon}{2k^2d}$ . Thus, the claim follows by union bound over all pairs of artificial-clusters and the setting of  $t$ .  $\triangleleft$

We are now ready to prove the correctness of the global algorithm, as stated in the next theorem.

$\blacktriangleright$  **Theorem 28.** *Under the promise that the input graph,  $G = (V, E)$ , is a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph, with clusters of size at least  $\beta n$ , Algorithm 2 computes a sparse spanning subgraph of  $G$ ,  $G' = (V, E')$ , such that:*

1. *The attained stretch is  $\Theta\left(\frac{\log n}{\phi_{\text{in}}^2}\right)$ , and*
2.  *$|E'| \leq n(1 + \varepsilon)$  with high probability.*

**Proof.** Let  $G = (V, E)$  be a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph and let  $C_1, \dots, C_f$  be a partition of  $G$  into  $f \leq k$   $(\phi_{\text{in}}, \phi_{\text{out}})$ -clusters of size at least  $\beta n$ . We begin by proving Item 1 of the claim. Let  $\mathcal{E}_1$  denote the event that for every  $i \in [f]$ ,  $C_i \cap \mathcal{S}$  contains a useful vertex w.r.t.  $C_i$ . Fix  $i \in [f]$ . By definition (see Claim 23), at least half of the vertices in  $C_i$  are useful. Since  $|C_i| \geq \beta n$ , the probability that  $v$  which selected u.a.r. from  $V$  is a useful vertex of  $C_i$  is at least  $\beta/2$ . Therefore w.p. at least  $1 - 1/n^c$ ,  $\mathcal{P}$  contains a useful vertex w.r.t.  $C_i$  where  $c$  is determined by the exact setting of  $p$ . Thus, by union bound over all  $i \in [f]$ , w.h.p.  $\mathcal{E}_1$  occurs.

Let  $\mathcal{E}_2$  denote the event that for every  $v \in V$ ,  $\Gamma_h(v) \cap \mathcal{S} \neq \emptyset$ . Conditioned on  $\mathcal{E}_1$ , by Claim 26 w.h.p.  $\mathcal{E}_2$  occurs.

Consider an edge  $\{u, v\}$  which belongs to  $E$  but not to  $E'$ . We first note that it follows that both  $\sigma(u) \neq \perp$  and  $\sigma(v) \neq \perp$  because otherwise  $\{u, v\}$  is added to  $E'$  in Step 9. From Claim 17 it follows that for every  $v \in V$  such that  $\sigma(v) \neq \perp$  the distance from  $v$  to  $\sigma(v)$  is at

most  $\frac{\log h}{\log(1+\phi_{\text{in}})}$ . To see this, observe that the  $\ell$ -ball centered at  $v$  in  $G[C]$  is contained  $\ell$ -ball centered at  $v$  in  $G$  where  $C$  denotes the cluster of  $v$  according to the partition mentioned above (in other words the neighborhood of  $v$  expands in  $G$  at least as fast as it does in  $G[C]$ ). Thus, if  $u$  and  $v$  belong to the same Voronoi cell, then the distance between them in  $G'$  is at most  $2\frac{\log h}{\log(1+\phi_{\text{in}})}$ . If  $u$  and  $v$  belong to the same artificial-cluster then the distance in  $G'$  between their respective centers,  $\sigma(u)$  and  $\sigma(v)$  is at most  $2\tau$  (because the distance in  $G'$  between a secondary center to its primary center is at most  $\tau$ ). Thus, in this case the distance in  $G'$  between  $u$  and  $v$  is at most  $2 \cdot \left(\tau + \frac{\log h}{\log(1+\phi_{\text{in}})}\right)$ . Finally, if  $u$  and  $v$  belong to different artificial-clusters then by Steps 10 and 11 there exists  $\{u', v'\} \in E'$  such that  $u'$  and  $u$  are in the same artificial-cluster and likewise for  $v$  and  $v'$ . Thus, by the above the distance in  $G'$  between  $u$  and  $v$  is at most  $2 \cdot \left(2 \cdot \left(\tau + \frac{\log h}{\log(1+\phi_{\text{in}})}\right)\right) + 1$ , where  $\tau \stackrel{\text{def}}{=} \frac{\log(2n^{3/2})}{\log\left(\left(1 - \frac{\phi_{\text{in}}^2}{2}\right)^{-1}\right)}$ .

Since for all  $x \geq 0$  it holds that  $x - x^2/2 \leq \ln(1+x)$  and since w.l.o.g.  $h \leq n$ , it follows that the attained stretch is  $\Theta\left(\frac{\log n}{\phi_{\text{in}}^2}\right)$ , as claimed.

Item 2 of the claim follows from the construction. More specifically, at Step 4 we add at most  $p \cdot r \cdot \tau = \Theta(\phi_{\text{out}}^{-1} \beta^{-1} \log^2 n)$  edges to  $E'$ . This is  $o(n)$  since we may assume that our query complexity is  $\tilde{O}(n^{2/3}) = o(n)$  (otherwise we may run the algorithm by Lenzen and Levi [12]). In Step 8, we add at most  $n - 1$  edges due to the fact that the Voronoi cells are vertex-disjoint. Conditioned on  $\mathcal{E}_2$ , which occurs w.h.p., in Step 9 we do not add any edges to  $E'$ . In Step 10 we add at most  $t$  edges to  $E'$ . Let  $\mathcal{E}_3$  denote the event that  $\mathcal{T}$  contains an edge from the edge-cut of every pair of artificial-clusters. By Claim 27 w.h.p.  $\mathcal{E}_3$  occurs. Finally, Conditioned on  $\mathcal{E}_3$ , in Step 11 we add at most  $k^2 \cdot \varepsilon \cdot n / (2k^2) = \varepsilon n / 2$  edges to  $E'$ . The claim follows.  $\blacktriangleleft$

### 4.3 Details of the Local Algorithm

In this section, we describe how to implement Algorithm 2 locally. The local implementation of all the steps of the algorithm is similar to the local implementation of the analogous steps of Algorithm 1 (with different parameters). The only new ingredient in the local implementation is the implementation of Step 11. Next, we describe how this step can be implemented locally. On query  $\{u, v\}$ , if  $u$  and  $v$  belong to the same artificial-cluster, then we proceed as before. Namely, we return YES if and only if  $u$  and  $v$  belong to the same Voronoi cell and  $\{u, v\}$  belongs to the tree that spans the cell. If  $u$  and  $v$  belong to different artificial-clusters then we consider three cases. The first case we consider is when  $\{u, v\} \in \mathcal{T}$ , the sample of edges selected in Step 10. In this case, we would like to return YES on the query  $\{u, v\}$ . The second case is when  $\{u, v\} \notin \mathcal{T}$  but there exists  $\{u', v'\} \in \mathcal{T}$  such that  $u$  and  $u'$  are in the same artificial-cluster and likewise for  $v$  and  $v'$ . In this case, we would like to return NO. Otherwise, we would like to return YES. Therefore, we can decide if to return YES or NO if we find for each edge in  $\mathcal{T}$  the identities of the artificial-clusters (i.e., the IDs of the primary centers of the corresponding artificial-clusters) of its endpoints. This is accomplished as follows. For each one of the pairs,  $(v, i)$ , selected in Step 10, we perform a neighbor query to obtain the  $i$ -th neighbor of  $v$ . If such a neighbor exists then let us denote it by  $u$ . We then find  $\sigma(v)$  and  $\sigma(u)$  by making  $O(hd^2)$  queries. This also reveals to which artificial-cluster  $v$  belongs and likewise for  $u$ . This allows us to obtain the list of pairs of artificial-clusters that already have an edge from their cut in  $\mathcal{T}$ . Thus, on query  $\{u, v\}$  where  $u$  and  $v$  belong to different artificial-clusters which are not on that list, the algorithm will return YES. Overall, implementing this check requires  $O(thd^2)$  queries and  $\tilde{O}(thd^2)$  time. We conclude that the query complexity of the algorithm is dominated by the implementation of Step 4 which requires  $O(pr\tau) = O(\log^2 n / (\beta\phi_{\text{out}}))$  queries and Step 11 which requires  $O(thd^2) = O(\phi_{\text{in}}^{-1} \varepsilon^{-1} k^3 d^3 \phi_{\text{out}} \cdot n \log^2 n)$  queries.

In this next claim, we bound the number of random bits that our algorithm uses.

▷ **Claim 29.** Performing  $r$   $s$ -wise independent random walks from each  $\psi \in \mathcal{P}$  in  $(G)_{\text{reg}}$  can be implemented by using  $O(\log n(\log n + \tau \log d))$  random bits and time-complexity  $r \cdot \text{poly}(\tau, \log n)$ .

*Proof.* As claimed in the proof of Claim 22, performing a single random-walk in  $(G)_{\text{reg}}$  of length  $\tau$  from any vertex requires  $\tau \log(2d)$  random bits. If we choose a random function from a family of  $s$ -wise independent functions,  $\mathcal{H}_{\gamma, \beta} = \{h : \{0, 1\}^\gamma \rightarrow \{0, 1\}^\beta\}$ , where  $\gamma = \log(p \cdot r)$  is the number of bits that are required to indicate the index of the vertex in  $\mathcal{P}$  and the index of the walk in  $[r]$  and  $\beta = \tau \log(2d)$  is the number of bits that are required for performing the walk then we obtain that the total number of random bits that the algorithm uses for performing the walks is  $s \cdot \max\{\gamma, \beta\} = O(\log n(\log n + \tau \log d))$ . We obtain the desired result since performing a single random walk requires  $\text{poly}(\gamma, \beta, s) = \text{poly}(\tau, \log n)$  time. ◀

We are now ready to prove Theorem 5.

▶ **Theorem 5.** *There is an LSSG algorithm that given query access to a connected  $(k, \phi_{\text{in}}, \phi_{\text{out}})$ -clusterable graph  $G = (V, E)$ , where each cluster is of size at least  $\beta \cdot n$ , provides access to  $G' = (V, E')$  such that the following holds. (1) The graph  $G'$  is a connected subgraph of  $G$  and with high probability  $|E'| \leq n(1 + \varepsilon)$ . Moreover, the stretch factor of  $G'$  is  $\Theta\left(\frac{\log n}{\phi_{\text{in}}^2}\right)$ . (2) The query complexity of the algorithm is  $O(\log^2 n \cdot (\beta \phi_{\text{out}})^{-1} + n \log^2 n \cdot k^3 d^3 \phi_{\text{out}} (\varepsilon \phi_{\text{in}})^{-1})$ , and (3) the number of random bits it uses is  $O\left(\frac{\log d \cdot \log^2 n}{\phi_{\text{in}}^2}\right)$ , where  $d$  is a bound on the maximum degree of  $G$ .*

**Proof of Theorem 5.** The correctness of the algorithm, i.e., Item (1), follows from Theorem 19. The query complexity of the algorithm is analyzed in the description of the local implementation that appears above. Finally, by Claim 22 the local implementation of Algorithm 1 uses  $O(\log n(\log n + \tau \log d)) = O\left(\frac{\log d \cdot \log^2 n}{\phi_{\text{in}}^2}\right)$  random bits, as claimed. This concludes the proof of the theorem. ◀

To obtain Corollary 6 from Theorem 5, we observe that we can take the upper bound on  $\phi_{\text{out}}$  to be the maximum between  $\phi_{\text{out}}$  and  $1/\sqrt{n}$ . More specifically, if we consider  $k, d, \phi_{\text{in}}, \varepsilon$  and  $\beta$  to be constants then the query complexity is  $\tilde{O}(1/\phi_{\text{out}} + n\phi_{\text{out}})$ . Since  $1/\phi_{\text{out}} > n\phi_{\text{out}}$  only when  $\phi_{\text{out}} < 1/\sqrt{n}$  we only need to show that the query complexity in this case is  $\tilde{O}(\sqrt{n})$ . Indeed, since  $\phi_{\text{out}}$  is only an upper bound on the outer-conductance, we may take  $\phi_{\text{out}} = 1/\sqrt{n}$  in this case and obtain the desired complexity.

▶ **Corollary 6.** *There is an LSSG algorithm that given query access to a connected  $(\Theta(1), \Theta(1), \phi_{\text{out}})$ -clusterable graph  $G = (V, E)$ , where each cluster is of size at least  $\beta \cdot n$ , provides access to  $G' = (V, E')$  such that the following holds. (1) The graph  $G'$  is a connected subgraph of  $G$  and with high probability  $|E'| \leq n(1 + \varepsilon)$ . Moreover, the stretch factor of  $G'$  is  $\Theta(\log n)$ . (2) The query and time complexity of the algorithm is  $\tilde{O}(\sqrt{n} + \phi_{\text{out}} n)$ , and (3) the number of random bits it uses is  $O(\log^2 n)$ .*

---

## References

- 1 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. SIAM, 2012.

- 2 Rubi Arviv, Lily Chung, Reut Levi, and Edward Pyne. Improved local computation algorithms for constructing spanners. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 3 Amartya Shankha Biswas, Ruidi Cao, Edward Pyne, and Ronitt Rubinfeld. Average-case local computation algorithms. *arXiv preprint arXiv:2403.00129*, 2024.
- 4 Greg Bodwin and Henry Fleischmann. Spanning adjacency oracles in sublinear time. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.
- 5 Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, 625(195-199):110, 1970.
- 6 Artur Czumaj, Pan Peng, and Christian Sohler. Testing cluster structure of graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 723–732, 2015.
- 7 Guy Even, Moti Medina, and Dana Ron. Best of two local models: Centralized local and distributed local algorithms. *Information and Computation*, 262:69–89, 2018. doi: 10.1016/j.ic.2018.07.001.
- 8 Uriel Feige, Yishay Mansour, and Robert E Schapire. Learning and inference in the presence of corrupted inputs. *Journal of Machine Learning Research*, 40(2015), 2015.
- 9 Shayan Oveis Gharan and Luca Trevisan. Partitioning into expanders. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1256–1266. SIAM, 2014.
- 10 Oded Goldreich. Basic facts about expander graphs. *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 451–464, 2011.
- 11 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
- 12 Christoph Lenzen and Reut Levi. A centralized local algorithm for the sparse spanning graph problem. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 13 Reut Levi and Moti Medina. A (centralized) local guide. *Bulletin of the EATCS*, 122:60–92, 2017. URL: <http://eatcs.org/beatcs/index.php/beatcs/article/view/495>.
- 14 Reut Levi, Guy Moshkovitz, Dana Ron, Ronitt Rubinfeld, and Asaf Shapira. Constructing near spanning trees with few local inspections. *Random Structures & Algorithms*, 50(2):183–200, 2017.
- 15 Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Transactions on Algorithms (TALG)*, 11(3):1–13, 2015.
- 16 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.
- 17 Reut Levi, Dana Ron, and Ronitt Rubinfeld. Local algorithms for sparse spanning graphs. *Algorithmica*, 82(4):747–786, 2020.
- 18 Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. *Algorithmica*, 4(77):971–994, 2016.
- 19 Reut Levi and Nadav Shoshan. Testing hamiltonicity (and other problems) in minor-free graphs. 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 61:1–61:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.



- 20 Yishay Mansour, Aviad Rubinfeld, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I 39*, pages 653–664. Springer, 2012.
- 21 Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 260–273. Springer, 2013.
- 22 Guy Moshkovitz and Asaf Shapira. Decomposing a graph into expanding subgraphs. *Random Struct. Algorithms*, 52(1):158–178, 2018. doi:10.1002/RSA.20727.
- 23 Merav Parter, Ronitt Rubinfeld, Ali Vakilian, and Anak Yodpinyanee. Local computation algorithms for spanners. *Innovations in Theoretical Computer Science (ITCS)*, 2019.
- 24 Pan Peng. Robust clustering oracle and local reconstructor of cluster structure of graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2953–2972. SIAM, 2020.
- 25 Dana Ron Reut Levi and Ronitt Rubinfeld. A local algorithm for constructing spanners in minor-free graphs. *Klaus Jansen, Claire Mathieu, José DP Rolim, and Chris Umans, editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 7–9, 2016.
- 26 Ronitt Rubinfeld. Can we locally compute sparse connected subgraphs? In *Computer Science—Theory and Applications: 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings 12*, pages 38–47. Springer, 2017.
- 27 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Proceedings of The Second Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011.
- 28 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.
- 29 Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1):93–133, 1989.
- 30 Daniel Spielman. Spectral and algebraic graph theory. *Yale lecture notes, draft of December*, 4:47, 2019.
- 31 Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

## **A** Other Related Work

### **A.1** LCAs for Spanners

A desirable property for a spanning subgraph is that it also preserves, up to a predetermined multiplicative factor  $\alpha \geq 1$  (a.k.a the stretch factor), the pairwise distances of the vertices in the original graph. Such a spanning subgraph is called an  $\alpha$ -spanner. In [15, 25]  $\text{poly}(h \log(d)/\varepsilon)$ -spanners for minor-free graphs are presented. For general (bounded degree) graphs, the algorithm of [12] outputs an  $O(\log^2 n \cdot \text{poly}(d/\varepsilon))$ -spanner. Their result is later extended by Parter et al. [23], who presented an algorithm that constructs an  $O(k^2)$ -spanner, independent of both  $n$  and  $d$ , but has  $O(n^{1+1/k})$  edges. The query complexity of [23] is  $O(n^{2/3}d^4)$ , and is later improved by Arviv et al. [2] to  $O(n^{2/3}d^2)$ . A recent work by Biswas, Cao, Pyne, and Rubinfeld [3] presents several LCAs that receive random graphs (i.e., Erdős-Rényi or Preferential Attachment graph) as an input and gives access to a sparse spanner of that graph.

## A.2 Graph Clustering

In the Property Testing model, Czumaj et al. [6] introduce an algorithm for testing the clusterability of a graph. In the Property Testing model, an algorithm accepts (with constant probability) any graph that is  $(k, \phi)$ -clusterable and rejects graphs that are  $\varepsilon$  far from being  $(k, \phi^*)$ -clusterable, i.e.,  $\varepsilon dn$  edges are required to be either added or removed from the input graph so that it becomes  $(k, \phi^*)$ -clusterable, where  $\phi^* = O(\frac{\phi^2 \varepsilon^2}{\log n})$ . The query complexity of their algorithm is  $\tilde{O}(\sqrt{n} \cdot \text{poly}(\phi, k, 1/\varepsilon))$  and with a success probability of at least  $2/3$ . Peng [24] uses similar ideas as [6] to construct a clustering oracle that given an input graph that is  $\varepsilon$ -close from being  $(k, \phi, O(\frac{\varepsilon \phi}{k^3 \log n}))$ -clusterable gives w.h.p. query access to the adjacency list of a  $(k, \frac{\phi}{2}, O(\frac{\sqrt{\varepsilon} \phi^{1.5}}{k^3 \log n}))$ -clusterable graph with preprocessing and query complexity of  $O(\sqrt{n} \cdot \text{poly}(\frac{k \log n}{\phi \varepsilon}))$ , and with at most  $O(k \sqrt{\frac{\varepsilon}{\phi}} \cdot n)$  outliers (vertices that are not associated with any cluster).

## A.3 LCAs for Other Graph Problems

The model of *local computation algorithms* (LCA) (sometimes also referred to as The Centralized-Local model (CentLocal)) as used in this work, was defined by Rubinfeld et al. [27] (see also [1] and survey in [13]). Such algorithms for maximal independent set, hypergraph coloring,  $k$ -CNF, approximated maximum matching and approximated minimum vertex cover for bipartite graphs are given in [27, 1, 20, 21, 7, 18, 8].

## B Omitted Proofs of Section 2

**Proof of Coro. 8.** By Theorem 7, for any distribution vector  $\vec{p}$  it holds that  $\|\hat{A}^\tau \vec{p} - \vec{u}\|_1 \leq \sqrt{n} \cdot \alpha^{\frac{\log(2n^{3/2})}{\log(1/\alpha)}} = \frac{1}{2n}$ . If there exists an index  $i$  such that either  $(\hat{A}^\tau \vec{p})_i < \frac{1}{2n}$  or  $(\hat{A}^\tau \vec{p})_i > \frac{3}{2n}$  then  $|\left(\hat{A}^\tau \vec{p}\right)_i - \frac{1}{n}| > \frac{1}{2n}$ , which implies that  $\|\hat{A}^\tau \vec{p} - \vec{u}\|_1 > \frac{1}{2n}$ , in contradiction to the above Equation. ◀

Proof of Claim 11. Since  $(G)_{\text{reg}}$  is  $2d$ -regular it holds that  $\lambda_1 = 2d$ . Let  $\{\hat{\lambda}_i\}_i$  denote the eigenvalues of  $\hat{A}$  where  $\hat{A} = \frac{1}{2d} A((G)_{\text{reg}})$ . By Theorem 9, it follows that  $\hat{\lambda}_1 = 1 > \hat{\lambda}_2, \dots, \geq \hat{\lambda}_n \geq -1$ . Since  $\hat{A}_{i,j} \geq 0$  and  $\hat{A}_{j,j} \geq \frac{1}{2}$  it holds that the matrix  $M \stackrel{\text{def}}{=} 2\hat{A} - I$ , where  $I$  is the identity matrix, is non-negative. Hence,  $M$  corresponds to a weighted connected graph. Moreover, the eigenvalues of  $M$ ,  $\{\mu_i\}_i$  satisfy  $\mu_i = 2\hat{\lambda}_i - 1$ .<sup>11</sup> In particular,  $\mu_1 = 1$ . Thus, Theorem 9 applied on  $M$  implies that  $\mu_i \geq \mu_n \geq -1$  for all  $1 \leq i \leq n$ . Thus,  $2\hat{\lambda}_i - 1 \geq -1$  and hence  $\hat{\lambda}_i \geq 0$  for all  $1 \leq i \leq n$ . Since  $\lambda_i = 2d\hat{\lambda}_i$ , it follows that  $|\lambda_2| > |\lambda_n|$ . The claim follows. ◀

Proof of Claim 12. We first observe that  $\phi(G) = \phi((G)_{\text{reg}})$ . Since  $(G)_{\text{reg}}$  is  $2d$ -regular, by Cheeger's Inequality it holds that  $\lambda_2 \leq 2d \cdot \left(1 - \frac{\phi^2(G)}{2}\right)$ . Thus the claim follows from Claim 11. ◀

<sup>11</sup>To see why  $\mu_i = 2\hat{\lambda}_i - 1$ , multiply the eigenvector that corresponds to  $\hat{\lambda}_i$ ,  $\nu_i$ , by  $M$  which gives  $2\hat{\lambda}_i \nu_i - \nu_i$ .

**C** Omitted Proofs of Section 3

Proof of Claim 17. Let  $v$  be a vertex in  $G$ . By Equation (1), for any subset  $S \subseteq V$  of size at most  $|V|/2$  it holds that  $e(S, V \setminus S) \geq \phi \cdot d \cdot |S|$ . In particular, if  $S$  is the set of vertices of the  $j$ -ball centered at some vertex  $v$  then the  $j + 1$ -ball centered at  $v$  contains at least  $|S| + \phi|S| = (1 + \phi)|S|$  vertices. Thus, after exploring  $\ell$  layers of the BFS rooted at  $v$  at least one of the following holds: either we explored more than  $|V|/2$  vertices, or we explored at least  $(1 + \phi)^\ell$  vertices. Thus we explore at least  $x$  vertices for any  $\ell$  such that  $(1 + \phi)^\ell \geq x$ . The claim follows.  $\triangleleft$



# When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

Kuan Cheng ✉ 🏠 

Center on Frontiers of Computing Studies, School of Computer Science, Peking University, China

Minghui Ouyang ✉ 

School of Mathematical Sciences, Peking University, China

Chong Shangguan ✉ 🏠 

Research Center for Mathematics and Interdisciplinary Sciences, Shandong University, China

Frontiers Science Center for Nonlinear Expectations, Ministry of Education, Qingdao, China

Yuanting Shen ✉ 

Research Center for Mathematics and Interdisciplinary Sciences, Shandong University, China

---

## Abstract

---

Tanner codes are graph-based linear codes whose parity-check matrices can be characterized by a bipartite graph  $G$  together with a linear inner code  $C_0$ . Expander codes are Tanner codes whose defining bipartite graph  $G$  has good expansion property. This paper is motivated by the following natural and fundamental problem in decoding expander codes:

What are the sufficient and necessary conditions that  $\delta$  and  $d_0$  must satisfy, so that *every* bipartite expander  $G$  with vertex expansion ratio  $\delta$  and *every* linear inner code  $C_0$  with minimum distance  $d_0$  together define an expander code that corrects  $\Omega(n)$  errors in  $O(n)$  time?

For  $C_0$  being the parity-check code, the landmark work of Sipser and Spielman (IEEE-TIT'96) showed that  $\delta > 3/4$  is sufficient; later Viderman (ACM-TOCT'13) improved this to  $\delta > 2/3 - \Omega(1)$  and he also showed that  $\delta > 1/2$  is necessary. For general linear code  $C_0$ , the previously best-known result of Dowling and Gao (IEEE-TIT'18) showed that  $d_0 = \Omega(c\delta^{-2})$  is sufficient, where  $c$  is the left-degree of  $G$ .

In this paper, we give a near-optimal solution to the above question for general  $C_0$  by showing that  $\delta d_0 > 3$  is sufficient and  $\delta d_0 > 1$  is necessary, thereby also significantly improving Dowling-Gao's result. We present two novel algorithms for decoding expander codes, where the first algorithm is deterministic, and the second one is randomized and has a larger decoding radius.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Pseudorandomness and derandomization; Theory of computation  $\rightarrow$  Expander graphs and randomness extractors; Theory of computation  $\rightarrow$  Error-correcting codes; Mathematics of computing  $\rightarrow$  Coding theory; Mathematics of computing  $\rightarrow$  Combinatoric problems

**Keywords and phrases** expander codes, expander graphs, linear-time decoding

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.61

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2312.16087> [10]

**Funding** The research of Chong Shangguan is supported by the National Key Research and Development Program of China under Grant No. 2021YFA1001000, the National Natural Science Foundation of China under Grant Nos. 12101364 and 12231014, and the Natural Science Foundation of Shandong Province under Grant No. ZR2021QA005.

**Acknowledgements** M. Ouyang would like to thank Extremal Combinatorics and Probability Group (ECOPRO), Institute for Basic Science (IBS, Daejeon, South Korea) for hosting his visit at the end of 2023.



© Kuan Cheng, Minghui Ouyang, Chong Shangguan, and Yuanting Shen; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 61; pp. 61:1–61:23



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Graph-based codes are an important class of error-correcting codes that have received significant attention from both academia and industry. They have a long history in coding theory, dating back to Gallager's [19] celebrated *low-density parity-check codes* (LDPC codes for short). LDPC codes are a class of linear codes whose parity-check matrices can be characterized by low-degree (sparse) bipartite graphs, called *factor graphs*. Gallager analyzed the rate and distance of LDPC codes, showing that with high probability, randomly chosen factor graphs give rise to error-correcting codes attaining the Gilbert-Varshamov bound. He also presented an iterative algorithm to decode these codes from errors caused by a binary symmetric channel. Since the 1990s, LDPC codes have received increased attention due to their practical and theoretical performance (see [12, 14, 22, 35, 38, 42, 43]).

As a generalization of the LDPC codes, Tanner [49] introduced the so-called *Tanner codes*, as formally defined below. Let  $c, d, n$  be positive integers and  $L := [n]$ , where  $[n] = \{1, \dots, n\}$ . Given a  $(c, d)$ -regular bipartite graph  $G$  with bipartition  $V(G) = L \cup R$  and a  $[d, k_0, d_0]$ -linear code  $C_0$ <sup>1</sup>, the *Tanner code*  $T(G, C_0) \subseteq \mathbb{F}_2^n$  is the collection of all binary vectors  $x \in \mathbb{F}_2^n$  with the following property: for every vertex  $u \in R$ ,  $x_{N(u)}$  is a codeword of the inner code  $C_0$ , where  $N(u) \subseteq L$  is the set of neighbors of  $u$  and  $x_{N(u)} = (x_v : v \in N(u)) \in \mathbb{F}_2^d$  denotes the length- $d$  subvector of  $x$  with coordinates restricted to  $N(u)$ ; in other words,  $T(G, C_0) := \{x \in \mathbb{F}_2^n : x_{N(u)} \in C_0 \text{ for every } u \in R\}$ .

*Expander codes* are Tanner codes whose defining bipartite graphs have good expansion properties, namely, they are *bipartite expanders*. To be precise, for real numbers  $\alpha, \delta \in (0, 1]$ , a  $(c, d)$ -regular bipartite graph  $G$  with bipartition  $V(G) = L \cup R$  with  $L = [n]$  is called a  $(c, d, \alpha, \delta)$ -*bipartite expander* if for each subset  $S \subseteq L$  with  $|S| \leq \alpha n$ ,  $S$  has at least  $\delta c|S|$  neighbors in  $R$ , i.e.,  $|N(S)| := |\cup_{v \in S} N(v)| \geq \delta c|S|$ . As each  $S \subseteq L$  can have at most  $c|S|$  neighbors in  $R$ , being a  $(c, d, \alpha, \delta)$ -bipartite expander means that every bounded size subset in  $L$  has as many neighbors in  $R$  as possible, up to a constant factor.

Sipser and Spielman [46] studied the Tanner code  $T(G, C_0)$  with  $G$  being a bipartite expander and  $C_0$  being a parity-check code. For simplicity, let  $Par = \{(x_1, \dots, x_d) : \sum_{i=1}^d x_i = 0\}$  denote the parity-check code in  $\mathbb{F}_2^d$ . They remarkably showed that the expansion property of  $G$  can be used to analyze the minimum distance and the decoding complexity of  $T(G, Par)$ . Roughly speaking, they showed that for every bipartite expander  $G$  with sufficiently large *expansion ratio*  $\delta > 1/2$ ,  $T(G, Par)$  has minimum distance at least  $\alpha n$ , which further implies that  $T(G, Par)$  defines a class of *asymptotically good* codes. More surprisingly, they showed that if the expansion ratio is even larger, say  $\delta > 3/4$ , then for every such  $G$ ,  $T(G, Par)$  admits a linear-time decoding algorithm that corrects a linear number of errors in the adversarial noise model. Spielman [48] showed that expander codes can be used to construct asymptotically good codes that can be encoded and decoded both in linear time.

Besides the construction based on vertex expansion, [48] also provides a construction based on spectral expansion. This construction again inherits the general structure of Tanner code, i.e., it combines of an underlying bipartite graph and an inner code. The main difference is that the underlying graph is an edge-vertex incidence graph of a (non-bipartite) spectral expander. Spectral expander codes also have linear time encoding and decoding, and their (rate & distance) parameters are different from those of vertex expander codes. In this paper, we mainly focus on vertex expander codes. The reader is referred to references [5, 26, 34, 36, 37, 41, 47, 52, 53] for more details on spectral expanders.

<sup>1</sup> The reader is referred to Section 1.2 for basic definitions on graphs and codes.

Given the strong performance of expander codes, they have been of particular interest in both coding theory and theoretical computer science, and have been studied extensively throughout the years. For example, [23, 45] utilized expander codes to obtain near MDS codes with linear-time decoding. A line of research [2, 9, 16, 18, 44, 50, 51, 52] improved the distance analysis and decoding algorithm for expander codes in various settings. Very recently, a sequence of works applied expander codes on quantum LDPC and quantum Tanner code construction, finally achieving asymptotically good constructions and linear-time decoding [6, 7, 15, 17, 21, 24, 27, 29, 30, 31, 32, 33, 39, 40].

Given the discussion above, it is natural to suspect that the expansion ratio  $\delta$  plays a prominent role in analyzing the properties of  $T(G, Par)$ . More precisely, one can formalize the following question. We always assume  $c, d, \alpha, \delta$  are constants while  $n$  tends to infinity.

► **Question 1.** *What is the minimum  $\delta > 0$  such that every  $(c, d, \alpha, \delta)$ -bipartite expander  $G$  with  $V(G) = L \cup R$  and  $|L| = n$  defines an expander code  $T(G, Par) \subseteq \mathbb{F}_2^n$  that corrects  $\Omega_{c,d,\alpha,\delta}(n)$  errors in  $O_{c,d,\alpha,\delta}(n)$  time?*

This question has already attracted considerable attention. Sipser and Spielman [46] used the bit-flipping algorithm (developed on the original algorithm of Gallager [19]) to show that  $\delta > 3/4$  is sufficient to correct  $(2\delta - 1)\alpha n$  errors in  $O(n)$  time. Using linear programming decoding, Feldman, Malkin, Servedio, Stein and Wainwright [18] showed that  $\delta > \frac{2}{3} + \frac{1}{3c}$  sufficient to correct  $\frac{3\delta-2}{2\delta-1}\alpha \cdot n$  errors, while at the cost of a poly( $n$ ) decoding time. Viderman [50] introduced the “Find Erasures and Decode” algorithm to show that  $\delta > \frac{2}{3} - \frac{1}{6c}$  is sufficient to correct  $\Omega(n)$  errors in  $O(n)$  time. Moreover, he also shows that there exists a  $(c, d, \alpha, 1/2)$ -bipartite expander  $G$  such that  $T(G, Par)$  only has minimum distance two, and therefore cannot correct even one error. Viderman’s impossibility result implies that  $\delta > 1/2$  is necessary for the assertion of Question 1 holding for *every*  $(c, d, \alpha, \delta)$ -bipartite expander.

The above results only consider the case where the inner code  $C_0$  is a parity-check code. Therefore, it is tempting to think about whether one can benefit from a stronger inner code  $C_0$ . Let us call a code *good* if it can correct  $\Omega(n)$  errors in  $O(n)$  time. Chilappagari, Nguyen, Vasic and Marcellin [11] showed that if  $G$  has expansion ratio  $\delta > 1/2$  and  $C_0$  has minimum distance  $d(C_0) \geq \max\{\frac{2}{2\delta-1} - 3, 2\}$ , then every such Tanner code  $T(G, C_0)$  is good. The above result implies that for  $\epsilon \rightarrow 0$  and  $\delta = 1/2 + \epsilon$ ,  $d(C_0) = \Omega(\epsilon^{-1})$  is sufficient to make every Tanner code  $T(G, C_0)$  good. Very recently, Dowling and Gao [16] significantly relaxed the requirement on  $\delta$  by showing that for every  $\delta > 0$ ,

$$d(C_0) \geq \Omega(c\delta^{-2}) \tag{1}$$

is sufficient<sup>2</sup> to make every Tanner code  $T(G, C_0)$  good, and be able to correct  $\alpha n$  errors. In particular, their result implies that, as long as the minimum distance of  $C_0$  is large enough, any tiny positive expansion ratio is sufficient to construct a good Tanner code.

Putting everything together, it is interesting to understand how the expansion ratio  $\delta$  of  $G$  and the minimum distance  $d_0$  of  $C_0$  affect the goodness of the Tanner code. We have the following generalized version of Question 1.

► **Question 2.** *What are the sufficient and necessary conditions that  $\delta$  and  $d_0$  must satisfy, so that every  $(c, d, \alpha, \delta)$ -bipartite expander  $G$  with  $V(G) = L \cup R$ ,  $|L| = n$ , and every inner linear code  $C_0 \subseteq \mathbb{F}_2^d$  with  $d(C_0) \geq d_0$ , together define an expander code  $T(G, C_0) \subseteq \mathbb{F}_2^n$  that corrects  $\Omega_{c,d,\alpha,\delta}(n)$  errors in  $O_{c,d,\alpha,\delta}(n)$  time?*

<sup>2</sup> More precisely,  $d(C_0) \geq 2t + c(t-1)^2 - 1$  with  $t > \frac{1}{\delta}$ .

The main purpose of this paper is to provide a near-optimal solution to the above question, as presented in the next subsection. On the negative side, we show that when  $d_0\delta \leq 1$ , there exists an extension of Videman's construction, yielding expander codes of constant distance (see Proposition 4 below). Therefore,  $d_0\delta > 1$  is a necessary condition for a general expander code  $T(G, C_0)$  to be considered good (compared to the  $\delta > \frac{1}{2}$  condition in the case of  $T(G, Par)$ ). On the positive side, we show that  $d_0\delta > 3$  is sufficient to make every expander code good (see Theorem 3 below for details). Our result only loses a multiplicity by three compared to the above necessary result.

## 1.1 Main results

### Deterministic decoding of expander codes

Our main result, which significantly improves on (1), is presented as follows.

► **Theorem 3.** *Let  $G$  be a  $(c, d, \alpha, \delta)$ -bipartite expander and  $C_0$  be a  $[d, k_0, d_0]$ -linear code, where  $c, d, \alpha, \delta, d_0, k_0$  are positive constants. If  $\delta d_0 > 3$ , then there exists a linear-time decoding algorithm for the Tanner code  $T(G, C_0)$  that can correct  $\gamma n$  errors, where  $\gamma = \frac{2\alpha}{d_0(1+0.5c\delta)}$ .*

Theorem 3 shows that  $\delta d_0 > 3$  is sufficient to make every Tanner code  $T(G, C_0)$  good. On the other hand, the next proposition shows that for every  $d_0 \geq 2$ ,  $\delta d_0 > 1$  is necessary.

► **Proposition 4.** *For every  $d, d_0 \geq 2$  and  $n \geq 10d_0$ , there exist constants  $0 < \alpha < 1, c \geq 3$  and a  $(c, d, 0.9\alpha, \frac{1}{d_0})$ -bipartite expander  $G$  with  $V(G) = L \cup R$  and  $|L| = n$  such that for every  $[d, k_0, d_0]$ -linear code  $C_0$ ,  $T(G, C_0)$  has minimum Hamming distance at most  $d_0$ .*

Theorem 3 and Proposition 4 together show that our requirement  $\delta d_0 = \Omega(1)$  is in fact almost optimal for Question 2. Moreover, we have the following conjecture on the fundamental trade-off between  $\delta$  and  $d_0$ .

► **Conjecture 5.** *If  $\delta d_0 > 1$ , then for every  $(c, d, \alpha, \delta)$ -bipartite expander  $G$  and every inner code  $C_0 \subseteq \mathbb{F}_2^d$  with  $d(C_0) \geq d_0$ , the expander code  $T(G, C_0) \subseteq \mathbb{F}_2^n$  can correct  $\Omega_{c,d,\alpha,\delta}(n)$  errors in  $O_{c,d,\alpha,\delta}(n)$  time.*

Due to space limit, we will omit the proof of the linear-running time of our algorithm, as well as the proof of Proposition 4. They can be found in the full version of this paper [10].

### Randomized decoding of expander codes

Another important direction in the study of expander codes is to understand the maximum number of errors that can be corrected in a linear-time decoding algorithm. Chen, Cheng, Li, and Ouyang [9] obtained a quite satisfactory answer to this problem for  $T(G, Par)$ . They showed that for every  $\delta > 1/2$  and  $(c, d, \alpha, \delta)$ -bipartite expander  $G$ ,  $T(G, Par)$  has minimum distance at least  $\frac{\alpha}{2(1-\delta)} \cdot n - O(1)$ , and this is tight up to a  $1 - o(1)$  factor. Moreover, for  $\delta > \frac{3}{4}$ , they also gave a linear-time decoding algorithm which corrects  $\frac{3\alpha}{16(1-\delta)} \cdot n$  errors. A similar problem for general expander codes  $T(G, C_0)$  was studied by [16].

Our decoding algorithm for Theorem 3 is deterministic and corrects  $\gamma n$  errors in linear time. Theorem 6 shows that one can correct more errors by using a randomized algorithm.

► **Theorem 6.** *Let  $G$  be a  $(c, d, \alpha, \delta)$ -bipartite expander and  $C_0$  be a  $[d, k_0, d_0]$ -linear code, where  $c, d, \alpha, \delta, d_0, k_0$  are positive constants. If  $\delta d_0 > 3$ , then there exists a linear-time randomized decoding algorithm for Tanner code  $T(G, C_0)$  such that if the input has at most  $\alpha n$  errors from a codeword, then with probability  $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$ , the decoding algorithm can output the correct codeword.*



## 1.2 Notations and definitions

A graph is a pair  $G = (V, E)$ , where  $V$  is a set whose elements are called vertices and  $E$  is a set of 2-subsets of  $V$ , whose elements are called edges. For a vertex  $u \in V$ , the set of *neighbors* of  $u$  in  $G$  is denoted by  $N(u) := \{v \in V : \{u, v\} \in E\}$ . For a subset  $S \subseteq V(G)$ , let  $N(S) = \cup_{u \in S} N(u)$  be the set of all the neighbors of the vertices in  $S$ . A graph  $G$  is *bipartite* if  $V(G)$  admits a bipartition  $V(G) = L \cup R$  such that both  $L$  and  $R$  contain no edge. Furthermore,  $G$  is  $(c, d)$ -regular if every vertex  $v \in L$  has exactly  $c$  neighbors in  $R$  and every vertex  $u \in R$  has exactly  $d$  neighbors in  $L$ .

Let  $\mathbb{F}_2 = \{0, 1\}$  denote the finite field of size 2. A code  $C$  is simply a subset of  $\mathbb{F}_2^n$ . For two vectors  $x = (x_1, \dots, x_n)$ ,  $y = (y_1, \dots, y_n) \in \mathbb{F}_2^n$ , the *Hamming distance* between  $x$  and  $y$ , denoted by  $d_H(x, y)$ , is the number of coordinates where  $x$  and  $y$  differ, that is,  $d_H(x, y) = |\{i \in [n] : x_i \neq y_i\}|$ . The *minimum distance* of a code  $C \subseteq \mathbb{F}_2^n$ , denoted by  $d(C)$ , is the minimum of  $d_H(x, y)$  among all distinct  $x, y \in C$ . Let  $\text{wt}(x)$  denote the number of nonzero coordinates of  $x$ . A code  $C \subseteq \mathbb{F}_2^n$  is said to be an  $[n, k, d(C)]$ -linear code if it is a linear subspace in  $\mathbb{F}_2^n$  with dimension  $k$  and minimum distance  $d(C)$ . It is well-known that for every linear code  $C$ ,  $d(C) = \min\{\text{wt}(x) : x \in C \setminus \{0\}\}$ .

Throughout, let  $G$  be a  $(c, d, \alpha, \delta)$ -bipartite expander, and  $C_0$  be a  $[d, k_0, d_0]$  linear code. Let  $T(G, C_0)$  be the Tanner code defined by  $G$  and  $C_0$ . Let *Check* be the error-detection algorithm of  $C_0$ , which checks whether a vector in  $\mathbb{F}_2^d$  is a codeword of  $C_0$ . Assume that *Check* takes  $h_0$  time. Similarly, let *Decode* be the correct-correction algorithm for  $C_0$ , which corrects up to  $\lfloor \frac{d_0-1}{2} \rfloor$  errors. Assume that *Decode* takes  $t_0$  time. Note that  $h_0, t_0$  are constants depending only on  $C_0$  but not on  $n$ .

Conventionally speaking, let us call the vertices in  $L$  *variables* and the vertices in  $R$  *constraints*. Given a vector  $x \in \mathbb{F}_2^n$ , which is corrupted from some codeword  $y \in T(G, C_0)$ , let us call a constraint  $u \in R$  *satisfied* if  $x_{N(u)} \in C_0$ , otherwise call it *unsatisfied*.

## 1.3 Some related works

Below, we briefly review two previous works [16, 46] that are closely related to our decoding algorithms for Theorems 3 and 6. Let us start from the decoding algorithm of Sipser and Spielman [46]. We summarize as follows the so-called iterated decoding or message-passing algorithm of [46] that decodes  $T(G, Par)$ .

- Let  $y \in T(G, Par)$  be the correct codeword that we want to decode from the received vector  $x$ . In the first round, the algorithm runs  $\text{Check}(x_{N(u)})$  for every  $u \in R$ . If a constraint  $u$  is unsatisfied, then it sends a “flip” message to every variable in  $N(u) \subseteq L$ . Sipser and Spielman showed that as long as the expansion ratio of  $G$  is sufficiently large ( $\delta > 3/4$ ) and the number of corruptions in  $x$  is sufficiently small but not identically zero (that is,  $1 \leq d_H(x, y) \leq (2\delta - 1)\alpha \cdot n$ ), then there must exist a variable  $v \in L$  that receives  $> c/2$  flip messages, which implies that more than half constraints in  $N(v)$  are unsatisfied. The algorithm then flips  $x_v$  and updates  $x$  and the status of the constraints in  $N(v)$ . Note that since  $Par$  is the parity-check code, flipping  $x_v$  makes all satisfied constraints in  $N(v)$  unsatisfied and all unsatisfied constraints in  $N(v)$  satisfied. Therefore, by flipping  $x_v$  one can strictly reduce the number of unsatisfied constraints.
- The algorithm then runs the above process repeatedly. As long as there are still unsatisfied constraints, it finds the desired  $v \in L$  so that flipping  $x_v$  strictly reduces the number of unsatisfied constraints. As there are at most  $|R| = cn/d$  unsatisfied constraints, the above process must stop in  $O(n)$  rounds and therefore yields an  $O(n)$  time decoding algorithm.

Dowling and Gao [16] extend Sipser and Spielman’s algorithm from  $T(G, Par)$  to the more general setting  $T(G, C_0)$  by making use of the minimum distance of  $C_0$ . Their algorithm works for linear codes defined on any finite field, but we will describe it only for  $\mathbb{F}_2$ .

- The algorithm begins by setting a threshold  $t \leq \lfloor \frac{d_0-1}{2} \rfloor$  and then runs  $\text{Decode}(x_{N(u)})$  for every  $u \in R$ . If a constraint  $u \in R$  satisfies  $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$ , then it sends a “flip” message to every variable  $v \in N(u)$  with  $\text{Decode}(x_{N(u)})_v \neq x_v$ . Note that  $\text{Decode}(x_{N(u)}) \in \mathbb{F}_2^d$  is a codeword in  $C_0$ . The algorithm then flips all  $x_v$  for those  $v$  receiving at least one flip, and then updates  $x$ . [16] showed that as long as the minimum distance  $d_0$  of  $C_0$  is sufficiently large (see (1)), then flipping all variables that receive at least one flip can reduce the number of corrupted variables in  $x$  by some positive fraction.
- In the next steps, the algorithm runs the above process repeatedly. As the number of corrupted variables is at most  $O(n)$ , the algorithm will stop in  $O(\log n)$  rounds. Crucially, in order to show that the running time of the algorithm is still linear-order but not of order  $n \log n$ , the authors proved that the running time of every single round is within a constant factor of the number of corrupted variables at the beginning of this round. As the numbers of corrupted variables form a decreasing geometric sequence with the leading term at most  $n$ , the total running time, which is within a constant factor of the sum of this geometric sequence, is also  $O(n)$ .

Lastly, it is worth mentioning that there are several very recent works on the explicit constructions of bipartite graphs with good vertex expansion properties, including the lossless expanders [8, 13, 20] and the unique-neighbor expanders [1, 3, 4, 25, 28].

#### 1.4 Key new ideas in our work

In this subsection, we briefly introduce the key new ideas in our work. Let us focus on the deterministic decoding algorithm that proves Theorem 3. Let us begin by analyzing the following two possible places where the previous algorithm in [16] could be improved.

In every decoding round of the above algorithm, the constraints in  $R$  which satisfy  $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$  (and hence send at least one and at most  $t-1$  flips to  $L$ ) in fact have two statuses, as detailed below. Let  $A$  be the set of constraints  $u \in R$  that sends at least one flip and  $\text{Decode}(x_{N(u)})$  computes the correct codeword in  $C_0$  (i.e.,  $\text{Decode}(x_{N(u)}) = y_{N(u)}$ ); let  $B$  be the set of constraints  $u \in R$  that sends at least one flip and  $\text{Decode}(x_{N(u)})$  computes an incorrect codeword in  $C_0$  (i.e.,  $\text{Decode}(x_{N(u)}) \neq y_{N(u)}$ ).

##### Two possible places where the previous algorithm could be improved

- (i) It could be the case that every constraint  $u \in A$  satisfies  $d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) = 1$  and hence sends only one correct flip to  $L$ ; in the meanwhile, every constraint  $u \in B$  may satisfy  $d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) = t-1$  and sends as many as  $t-1$  flips to  $L$ , which could be all wrong. In this case, the constraints in  $R$  altogether send  $|A|$  correct flips and  $(t-1)|B|$  wrong flips to the variables in  $L$ .
- (ii) Unfortunately, the situation could be worse. Since our bipartite graph  $G$  is  $(c, d)$ -regular, it could be the case that the neighbors of the constraints in  $A$  are highly concentrated (e.g., all  $|A|$  correct flips are received by as few as  $|A|/c$  variables in  $L$ ), and the neighbors of the constraints in  $B$  are highly dispersed (e.g., all  $(t-1)|B|$  possibly wrong flips are received by as many as  $(t-1)|B|$  variables in  $L$ ). Consequently, a small number of corrupted variables but a large number of correct variables in  $L$  receive flip messages.

Given the two issues above, if we flip all variables that receive at least one flip, then in the worst case, we could correct  $|A|/c$  old corrupt variables but produce  $(t-1)|B|$  new corrupt variables. Recall that to make the algorithm in [16] work, in each round, we need

to reduce the number of corrupted variables by at least a positive fraction, which implies that in the worst case it is necessary to have  $|A|/c \geq (t-1)|B|$ . Together with some lower bound on  $|A|$  and upper bound on  $|B|$  (see [16] for details), one can prove that in such worst scenario (1) is necessary for Dowling and Gao's algorithm to work.

Our new algorithm begins by noting that we could indeed fix the two problems mentioned above. To do so, we introduce several new ideas as briefly presented below.

### Key new ideas in our work

Let  $F := \{i \in [n] : x_i \neq y_i\}$  be the set of corrupt variables in  $x$ . Similarly to [16], our new algorithm begins by setting a threshold  $t = \lfloor \frac{1}{\delta} \rfloor$  and then runs  $\text{Decode}(x_{N(u)})$  for every  $u \in R$ .

- (a) To fix the first problem, if a constraint  $u \in R$  satisfies  $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t-1$ , then instead of sending a flip message to every  $v \in N(u)$  with  $\text{Decode}(x_{N(u)})_v \neq x_v$ , the new algorithm just arbitrarily picks *exactly one* such variable  $v$ , and sends a flip message to *only* this specific  $v$ . So, every constraint in  $A \cup B$  sends exactly one flip to  $L$ .
- (b) To fix the second problem, we associate each  $v \in L$  with a counter  $\tau_v \in \{0, 1, \dots, c\}$  that counts the number of flips received by  $v$ . For each  $m \in [c]$ , let  $S_m$  denote the set of variables that receive exactly  $m$  flips. Then, instead of flipping every variable that receives at least one flip, i.e., instead of flipping  $\cup_{m=1}^c S_m$ , we only flip  $S_m$  for some  $m \in [c]$ . Crucially, we show that if the number  $|F|$  of corrupt variables is not too large, then there must *exist* some  $m \in [c]$  such that  $|S_m|$  has the same order as  $|F|$ , and more importantly, a  $(1/2 + \kappa)$ -fraction of variables in  $|S_m|$  are corrupted (and therefore can be corrected by the flipping operation), where  $\kappa$  is an absolute positive constant. Therefore, it follows that by flipping all variables in  $S_m$ , one can reduce  $|F|$  by some positive fraction.

Note that the details of (a) and (b) can be found in Section 3.2, where we call the algorithm corresponding to (a) and (b) “EasyFlip” and write  $\text{EasyFlip}(x, m)$  as the output of the EasyFlip if  $x$  is the input vector and  $S_m$  is flipped (see Algorithm 2).

However, there is still a gap that needs to be fixed, that is, how to find the required  $S_m$ ? A plausible solution is to run  $\text{EasyFlip}(x, m)$  for every  $m \in [c]$ . This would roughly increase the total running time by a  $c$  factor, which will still be  $O(n)$ , provided that the original running time is  $O(n)$ . Unfortunately, by doing so we still cannot *precisely* identify the required  $S_m$ , as in general we do not know how to count the number of corrupted variables in some corrupted vector. We will fix this issue by introducing our third key new idea:

- (c) Note that what we can explicitly count in each round of the algorithm is the number of unsatisfied constraints. Roughly speaking, our strategy is to run EasyFlip iteratively for a large but still constant number of times and then pick the final output that significantly reduces the number of unsatisfied constraints.

More precisely, assume that we will run EasyFlip iteratively for  $s$  rounds. Let  $x^0 := x$  and write  $x^1 := \text{EasyFlip}(x^0, m_1)$  as the output of the 1st EasyFlip invocation where the variables in  $S_{m_1}$  are flipped for some  $m_1 \in [c]$ ; more generally, for  $k \in [s]$ , write  $x^k := \text{EasyFlip}(x^{k-1}, m_k)$  as the output of the  $k$ th EasyFlip invocation where the variables in  $S_{m_k}$  is flipped for some  $m_k \in [c]$ . Note that in Algorithm 3 we call the above iterated invocations of EasyFlip as “DeepFlip”, and write  $x^k := \text{DeepFlip}(x, (m_1, \dots, m_k))$  as the output of the  $k$ th EasyFlip invocation. For  $0 \leq k \leq s$ , let  $F^k \subseteq L$  and  $U^k \subseteq R$  denote the sets of corrupted variables and unsatisfied constraints caused by  $x^k$ , respectively. We prove that there are constants  $0 < \epsilon \ll \epsilon' \ll \epsilon'' < 1$  such that the following two wordy but useful observations hold:

- (c1) If the number of corrupted variables is reduced *dramatically* then the number of unsatisfied constraints is reduced *significantly*, i.e., if for some  $k \in [s]$ ,  $|F^k| \leq \epsilon|F^0|$ , then  $|U^k| \leq \epsilon'|U|$ ;
- (c2) If the number of unsatisfied constraints is reduced *significantly*, then the number of corrupted variables must be reduced by a least a constant fraction i.e., if for some  $k \in [s]$ ,  $|U^k| \leq \epsilon'|U^0|$ , then  $|F^k| \leq \epsilon''|F|$ .

In the following, we will briefly argue how we will make use of the two observations (c1) and (c2). Recall that in (b) we have essentially guaranteed that for every  $k \in [s]$ , there exists some  $m_k^* \in [c]$  such that by flipping  $S_{m_k^*}$  in EasyFlip, one could reduce the number of corrupted variables by an  $\eta$ -fraction for some  $\eta \in (0, 1)$ . It follows that if we run DeepFlip iteratively for  $(m_1, \dots, m_s) = (m_1^*, \dots, m_s^*)$ , then we have  $|F^s| \leq (1 - \eta)^s |F| < \epsilon|F|$ , provided that  $s > \log_{(1-\eta)^{-1}} \epsilon^{-1}$  is sufficiently large (but still a constant independent of  $n$ ). Therefore, if we run DeepFlip thoroughly for all  $(m_1, \dots, m_s) \in [c]^s$ , then by (c1) there must exist at least one<sup>3</sup>  $x^k := \text{DeepFlip}(x, (m_1, \dots, m_k))$  with  $k \leq s$  such that  $|U^k| \leq \epsilon'|U|$ . Moreover, using the last inequality, such  $x^k$  and  $(m_1, \dots, m_k)$  can be explicitly identified. Now, by (c2) we can conclude that the number of corrupted variables is indeed reduced by at least a constant fraction.

Note that the above brute-force search only increases the total running time by at most a  $c^s$  factor. The details of (c) and the analysis of DeepFlip can be found in Section 3.3 and Algorithm 3. Moreover, we call the algorithm that runs  $\text{DeepFlip}(x, (m_1, \dots, m_s))$  thoroughly for all  $(m_1, \dots, m_s) \in [c]^s$  as “HardSearch”, and is discussed in Section 3.4 and Algorithm 4. The discussion above basically shows that every HardSearch invocation could reduce the number of corrupted variables by a constant fraction.

Running HardSearch iteratively for  $O(\log n)$  rounds, the total number of corrupted variables will be smaller than  $\lfloor \frac{d_0-1}{2} \rfloor$ , which can be easily corrected by running Decode for every  $u \in R$ . The main algorithm that puts everything together is called “MainDecode”, and is presented in Section 3.1 and Algorithm 1.

To show that the total running time is still linear in  $n$ , we adopt an argument similar to that in the previous works (e.g., [16]). We show that the running time of every HardSearch invocation is within a constant factor of the number of corrupted variables at the beginning of this invocation.

Lastly, we would like to mention that our randomized decoding algorithm (see Algorithm 5), which proves Theorem 6 and has a larger decoding radius than the deterministic algorithm, basically follows from the same framework mentioned above. Loosely speaking, the high-level idea of the randomized algorithm is to reduce the number of corruptions to a moderate size that can be handled by the deterministic algorithm. For that purpose, we design a random flip strategy which can be summarized as follows.

Recall that for every  $m \in [c]$ ,  $S_m$  denotes the set of all variables that receive exactly  $m$  flips. First, for every constraint  $u \in R$  satisfying  $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t$ , we arbitrarily pick exactly one variable  $v \in N(u)$  with  $\text{Decode}(x_{N(u)})_v \neq x_v$  and send a flip message to this specific  $v$ . Then, we collect all suspect variables that receive at least one flip. Subsequently, we design a random sampling procedure to select a subset of  $\cup_{m \in [c]} S_m$  to flip. We show that this procedure can ensure, with high probability, that this subset has more corrupted variables than correct variables, as long as the total number of corruptions is at most  $\alpha n$ . By applying this strategy iteratively, we can show that in each iteration, the number of corruptions will be reduced by a positive fraction. Then, after running a

<sup>3</sup> Clearly,  $\text{DeepFlip}(x, (m_1^*, \dots, m_s^*))$  gives a candidate for such  $x^k$ .

constant number of iterations, the number of corrupted bits can be reduced to a range that the deterministic algorithm can handle. At this point, the deterministic algorithm is invoked to correct all remaining corrupted variables. Moreover, as  $n$  increases, the fail probability of each iteration tends to 0. So the overall random algorithm will succeed with high probability.

## 2 An auxiliary lemma

Given two subsets  $S, T \subseteq V(G)$ , let  $E(S, T)$  denote the set of edges with one endpoint in  $S$  and another endpoint in  $T$ . For every positive integer  $t$ , let

- $N_{\leq t}(S) = \{u \in V(G) : 1 \leq |N(u) \cap S| \leq t\}$ ,
- $N_t(S) = \{u \in V(G) : |N(u) \cap S| = t\}$ ,
- and  $N_{\geq t}(S) = \{u \in V(G) : |N(u) \cap S| \geq t\}$ .

We will make use of the following crucial property of bipartite expander graphs.

► **Proposition 7 (Folklore).** *Let  $G$  be a  $(c, d, \alpha, \delta)$ -bipartite expander. Then, for every set  $S \subseteq L$  with  $|S| \leq \alpha n$  and every integer  $t \in [d]$ , we have that  $|N_{\leq t}(S)| \geq \frac{\delta(t+1)-1}{t} \cdot c|S|$ .*

## 3 Deterministic decoding: Decoding $\Omega(n)$ corruptions in $O(n)$ time

We need to set some parameters. Suppose that  $d_0 > \frac{3}{\delta} - 1$ . Let  $t = \lfloor \frac{1}{\delta} \rfloor$ . Take  $\epsilon_0 > 0$  such that  $d_0 > \frac{3}{\delta} - 1 + 2\epsilon_0$  and  $\lfloor \frac{1}{\delta} + \epsilon_0 \rfloor = \lfloor \frac{1}{\delta} \rfloor$ . For every  $0 < \epsilon_1 < \frac{\epsilon_0 \delta^2}{100}$ , let  $\epsilon_2 = \frac{\epsilon_1}{c+1} \cdot \frac{\delta(t+1)-1}{t} > 0$  and  $\epsilon_3 = \epsilon_2 \left( 2(1 - \epsilon_1) \left( \frac{1}{2} + \frac{\epsilon_0 \delta^2}{2} \right) - 1 \right) > 0$ . It is not hard to check that  $\epsilon_1, \epsilon_2$  and  $\epsilon_3$  are all well-defined. Lastly, let  $\epsilon_4 = \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1 - \epsilon_3)$ ,  $\ell = \left\lceil \log_{1-\epsilon_3} \left( \lfloor \frac{d_0-1}{2} \rfloor \frac{1}{\gamma n} \right) \right\rceil$  and  $s_0 = \left\lceil \log_{1-\epsilon_3} \left( \epsilon_4 \frac{\delta d_0 - 1}{d_0 - 1} \right) \right\rceil$ .

### 3.1 The main decoding algorithm – MainDecode

Given a corrupt vector  $x \in \mathbb{F}_2^n$  with at most  $\gamma n$  corruptions, our main decoding algorithm (see Algorithm 1 below) works as follows. The algorithm is divided into two parts. In the first part (see steps 2-10 below), it invokes `HardSearch` (see Algorithm 4 below) recursively for  $\ell$  rounds, where in every round the number of corrupt variables is reduced by a  $(1 - \epsilon_3)$ -fraction. After  $\ell$  executions of `HardSearch`, the number of corrupt variables is reduced to at most  $\lfloor \frac{d_0-1}{2} \rfloor$ . Then, in the second part of the algorithm (see steps 11-13 below), the decoder of the inner code  $C_0$  is applied to finish decoding.

The next two lemmas justify the correctness and the linear running time of `MainDecode`.

► **Lemma 8.**

- (i) *Let  $x$  be the input vector of `HardSearch` and let  $F$  be the set of corrupt variables of  $x$ . Let  $x' := \text{HardSearch}(x)$  and  $F'$  be the set of corrupt variables of  $x'$ . If  $|F| \leq \gamma n$ , then  $|F'| \leq (1 - \epsilon_3) \cdot |F|$ .*
- (ii) *In step 11 of `MainDecode`, the number of corrupt variables in  $x^\ell$  is at most  $\lfloor \frac{d_0-1}{2} \rfloor$ .*

► **Lemma 9.**

- (i) *Let  $x$  be the input vector of `HardSearch` and let  $F$  be the set of corrupt variables of  $x$ . If  $|F| \leq \gamma n$ , then the running time of `HardSearch` is at most  $O(n + |F|)$ .*
- (ii) *Furthermore, if the number of corrupt variables in the input vector of `MainDecode` is at most  $\gamma n$ , then the running time of `MainDecode` is  $O(n)$ .*

## 61:10 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

■ **Algorithm 1** Main decoding algorithm for expander codes – MainDecode.

---

**Input:**  $G, C_0, x \in \mathbb{F}_2^n$   
**Output:**  $x' \in \mathbb{F}_2^n$

- 1 Set  $i = 1$  and  $x^0 = x$ ;
- 2 **for**  $1 \leq i \leq \ell$  **do**
- 3      $U^{i-1} \leftarrow \{u \in R : x_{N(u)}^{i-1} \notin C_0\}$ ;
- 4     **if**  $|U^{i-1}| = 0$  **then**
- 5         return  $x' \leftarrow x^{i-1}$ ;
- 6     **else**
- 7          $x^i \leftarrow \text{HardSearch}(x^{i-1})$ ;
- 8          $i \leftarrow i + 1$  ;
- 9     **end**
- 10 **end**
- 11 **for every**  $u \in R$  **such that**  $x_{N(u)}^\ell \notin C_0$  **do**
- 12      $x_{N(u)}^\ell \leftarrow \text{Decode}(x_{N(u)}^\ell)$
- 13 **end**
- 14 return  $x' \leftarrow x^\ell$ ;

---

**Proof of Theorem 3.** Let  $y \in T(G, C_0)$  be a codeword and  $x \in \mathbb{F}_2^n$  be a corrupted vector. Let  $F = \{i \in [n] : x_i \neq y_i\}$  be the set of corrupt variables of  $x$  with respect to  $y$ . To prove the theorem, it suffices to show that as long as  $|F| \leq \gamma n$ , MainDecode finds  $y$  correctly in linear time. We will analyze the following two cases:

- If the algorithm returns  $x^i$  for some  $0 \leq i \leq \ell - 1$ , then as  $|U^i| = 0$ , we must have  $x^i \in T(G, C_0)$ . Let  $F^i$  be the set of the corrupt variables of  $x^i$ . Then it follows by Lemma 8 (i) that  $d(x^i, y) = |F^i| \leq (1 - \epsilon_3)^i |F| \leq (1 - \epsilon_3)^i \gamma n < d(T(G, C_0))$ , which implies that  $x^i = y$ .
- If the algorithm does not return  $x^i$  for any  $0 \leq i \leq \ell - 1$ , then it follows by Lemma 8 (ii) that  $d(x^\ell, y) \leq \lfloor \frac{d_0 - 1}{2} \rfloor$ . Therefore, one can find  $y$  by running Decode for every  $u \in R$ .

Moreover, by Lemma 9 the running time of MainDecode is  $O(n)$ . ◀

The remaining part of this section is organized as follows. In Section 3.2 below, we will introduce the basic building block of deterministic decoding – EasyFlip, which also corresponds to items (a) and (b) in Section 1.4. In Section 3.3 we will introduce the algorithm DeepFlip, which runs EasyFlip iteratively for a constant number of times. DeepFlip corresponds to item (c) in Section 1.4. In Section 3.4 we will introduce HardSearch, which is designed by running DeepFlip thoroughly for all choices of  $(m_1, \dots, m_s)$  until the number of unsatisfied constraints is significantly reduced. The proof of Lemma 8 is also presented in Section 3.4.

### 3.2 The basic building block of deterministic decoding – EasyFlip

In this subsection, we will present the algorithm EasyFlip (see Algorithm 2 below), which is the basic building block of our deterministic decoding. It contains the following two parts:

- EasyFlip (i): in the first part (see steps 1-6 below), it invokes Decode for each constraint  $u \in R$  and sends flips to some variables  $v \in L$ ;
- EasyFlip (ii): in the second part (see steps 7-11 below), it counts the number of flips received by each variable in  $L$  and flips all variables that receive exactly  $m$  flips.

■ **Algorithm 2** Flip all variables receiving exactly  $m$  flips – EasyFlip.

---

**Input:**  $G, C_0, x \in \mathbb{F}_2^n$ , and  $m \in [c]$   
**Output:**  $x' \in \mathbb{F}_2^n$

```

1 for every  $u \in R$  do
2    $\omega \leftarrow \text{Decode}(x_{N(u)});$ 
3   if  $1 \leq d_H(\omega, x_{N(u)}) \leq t$  then
4     send a “flip” to an arbitrary vertex  $v \in N(u)$  with  $\omega_v \neq x_v$ 
5   end
6 end
7 for every  $v \in L$  do
8   if  $v$  receives exactly  $m$  flips then
9     flip  $x_v$ 
10  end
11 end
12 return  $x' \leftarrow x$ 

```

---

Our goal is to show that there must exist an integer  $m \in [c]$  such that by flipping all variables  $v \in L$  that receive exactly  $m$  flips, one can reduce the number of corrupt variables in  $x'$  by a  $(1 - \epsilon_3)$ -fraction, as compared with  $x$ . Note that for this moment, it suffices to prove the existence of such an  $m$  and we do not need to find it explicitly. In fact, later we will find the required  $m$  by exhaustive search.

We make the discussion above precise by the following lemma.

► **Lemma 10.** *Let  $x$  be the input vector of EasyFlip and let  $F$  be the set of corrupt variables of  $x$ . If  $|F| \leq \alpha n$ , then there exists an integer  $m \in [c]$  such that the following holds. Let  $x' = \text{EasyFlip}(x, m)$  be the output vector of EasyFlip and  $F'$  be the set of corrupt variables of  $x'$ . Then  $|F'| \leq (1 - \epsilon_3)|F|$ .*

### 3.2.1 Proof of Lemma 10

Let us first introduce some notation and easy inequalities. Let  $y \in T(G, C_0)$  be the correct codeword that we want to decode from  $x$ . Let  $A$  be the set of constraints  $u \in R$  that sends a flip and  $\text{Decode}(x_{N(u)})$  computes the correct codeword in  $C_0$  (that is,  $\text{Decode}(x_{N(u)}) = y_{N(u)}$ ). Similarly, let  $B$  be the set of constraints  $u \in R$  that sends a flip and  $\text{Decode}(x_{N(u)})$  computes an incorrect codeword in  $C_0$  (i.e.,  $\text{Decode}(x_{N(u)}) \neq y_{N(u)}$ ).

By the definitions of  $A$  and  $N_{\leq t}(F)$ , it is easy to see that

$$A = \{u \in R : 1 \leq |N(u) \cap F| \leq t\} = N_{\leq t}(F). \quad (2)$$

Therefore, it follows by (2) and Proposition 7 that

$$|A| \geq \frac{\delta(t+1) - 1}{t} \cdot c|F|. \quad (3)$$

Moreover, since a constraint  $u \in R$  computes an incorrect codeword in  $C_0$  only if it sees at least  $d_0 - t$  corrupt variables in its neighbors (recall that  $d(C_0) \geq d_0$ ), we have that

$$B = \{u \in R : |N(u) \cap F| \geq d_0 - t \text{ and } \exists \omega \in C_0 \text{ s.t. } 1 \leq d_H(\omega, x_{N(u)}) \leq t\} \subseteq N_{\geq d_0-t}(F). \quad (4)$$

By counting the number of edges between  $F$  and  $N(F)$ , we see that

$$(d_0 - t)|B| \leq |E(F, B)| \leq |E(F, N(F))| = c|F|,$$

## 61:12 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

which implies that

$$|B| \leq \frac{c|F|}{d_0 - t}. \quad (5)$$

Consider the following two equalities,

$$\sum_{k=1}^d k \cdot |N_k(F)| = c|F| \quad \text{and} \quad \sum_{k=1}^d |N_k(F)| = |N(F)| \geq \delta c|F|.$$

By multiplying the second by  $\frac{1}{\delta} + \epsilon_0$  and subtracting the first one, we have

$$\sum_{k=1}^t \left(\frac{1}{\delta} + \epsilon_0 - k\right) |N_k(F)| - \sum_{k=t+1}^d \left(k - \frac{1}{\delta} - \epsilon_0\right) |N_k(F)| \geq \left(\left(\frac{1}{\delta} + \epsilon_0\right) \delta - 1\right) c|F| \geq \epsilon_0 \delta c|F|,$$

Moreover, it follows by (2) and (4) that

$$\begin{aligned} & \sum_{k=1}^t \left(\frac{1}{\delta} + \epsilon_0 - k\right) |N_k(F)| - \sum_{k=t+1}^d \left(k - \frac{1}{\delta} - \epsilon_0\right) |N_k(F)| \\ & \leq \sum_{k=1}^t \left(\frac{1}{\delta} + \epsilon_0 - k\right) |N_k(F)| - \sum_{k=d_0-t}^d \left(k - \frac{1}{\delta} - \epsilon_0\right) |N_k(F)| \\ & \leq \left(\frac{1}{\delta} + \epsilon_0 - 1\right) |N_{\leq t}(F)| - \left(d_0 - t - \frac{1}{\delta} - \epsilon_0\right) |N_{\geq d_0-t}(F)| \\ & \leq \left(\frac{1}{\delta} + \epsilon_0 - 1\right) |A| - \left(d_0 - t - \frac{1}{\delta} - \epsilon_0\right) |B|. \end{aligned}$$

As  $d_0 > \frac{3}{\delta} - 1 + 2\epsilon_0$  and  $t = \lfloor \frac{1}{\delta} \rfloor$ , we have  $d_0 - t - \frac{1}{\delta} - \epsilon_0 > \frac{1}{\delta} + \epsilon_0 - 1$ . Combining the above two inequalities, one can infer that

$$\epsilon_0 \delta c|F| \leq \left(\frac{1}{\delta} + \epsilon_0 - 1\right) |A| - \left(d_0 - t - \frac{1}{\delta} - \epsilon_0\right) |B| \leq \left(\frac{1}{\delta} + \epsilon_0 - 1\right) (|A| - |B|) \leq \frac{1}{\delta} (|A| - |B|),$$

which implies that

$$|A| - |B| \geq \epsilon_0 \delta^2 c|F|. \quad (6)$$

On the other hand, since  $A$  and  $B$  are disjoint subsets of  $N(F)$ , we have that

$$|A| + |B| \leq |N(F)| \leq c|F|. \quad (7)$$

For every integer  $m \in [c]$ , let  $S_m$  be the set of variables in  $L$  that receive exactly  $m$  flips. Then the variables in  $S_m$  receive a total number of  $m|S_m|$  flips. In EasyFlip, every constraint in  $A \cup B$  sends exactly one flip to  $L$ . The total number of flips sent by constraints in  $R$  and received by variables in  $L$  is exactly

$$|A| + |B| = \sum_{m=1}^c m|S_m|. \quad (8)$$

Let  $Z$  be the set of correct variables that receive at least one flip, i.e.,  $Z = (\cup_{m=1}^c S_m) \setminus F$ . Observe that the set  $F'$  of corrupt variables in the output vector  $x'$  consists of corrupt variables not flipped by EasyFlip, which is  $F \setminus S_m$ , and correct variables that are erroneously flipped by EasyFlip, which is  $S_m \cap Z$ . Therefore,

$$F' = (F \setminus S_m) \cup (S_m \cap Z). \quad (9)$$



Let  $\alpha_m$  be the fraction of corrupt variables in  $S_m$ . Then we have that

$$\alpha_m = \frac{|S_m \cap F|}{|S_m|} \quad \text{and} \quad 1 - \alpha_m = \frac{|S_m \cap Z|}{|S_m|}. \quad (10)$$

Let  $\beta_m$  denote the fraction of flips sent from  $A$  to  $S_m$  among all flips received by  $S_m$ , i.e.,

$$\beta_m = \frac{\text{the number of flips sent from } A \text{ to } S_m}{m|S_m|}. \quad (11)$$

The following inequality is crucial in the analysis of EasyFlip.

▷ **Claim 11.** For every  $m \in [c]$ ,  $\alpha_m \geq \beta_m$ .

*Proof.* As every variable in  $S_m$  receives the same number of  $m$  flips, by (10) the number of flips received by  $S_m \setminus F$  is  $(1 - \alpha_m)m|S_m|$ . Moreover, by (11) the number of flips sent from  $B$  to  $S_m$  is  $(1 - \beta_m)m|S_m|$ . Since the constraints in  $A$  always compute the correct codewords in  $C_0$ , they always send correct flips to their neighbors in  $L$ . Therefore, the flips received by  $S_m \setminus F$  (which are the wrong flips) must be sent by  $B$ , which implies that  $(1 - \alpha_m)m|S_m| \leq (1 - \beta_m)m|S_m|$ , where the inequality follows from the fact that  $B$  could also send flips to  $S_m \cap F$  (which are the correct flips). Thus,  $\alpha_m \geq \beta_m$ , as needed. ◁

The following result shows that there exists an integer  $m \in [c]$  such that there exists a large set  $S_m$  that contains many corrupt variables.

▷ **Claim 12.** If  $|F| \leq \alpha n$ , then there exists an integer  $m \in [c]$  such that  $\alpha_m \geq (1 - \epsilon_1) \frac{|A|}{|A| + |B|}$  and  $|S_m| \geq \epsilon_2 |F|$ .

*Proof.* Suppose for the sake of contradiction that for every  $m \in [c]$ , we have either  $\alpha_m < (1 - \epsilon_1) \frac{|A|}{|A| + |B|}$  or  $|S_m| < \epsilon_2 |F|$ . Then, by counting the number of flips sent from  $A$  to  $L$  (which is exactly  $|A|$ ), we have that

$$\begin{aligned} |A| &= \sum_{m=1}^c \beta_m m |S_m| \leq \sum_{m=1}^c \alpha_m m |S_m| < (1 - \epsilon_1) \frac{|A|}{|A| + |B|} \sum_{m=1}^c m |S_m| + \sum_{m=1}^c m \epsilon_2 |F| \\ &= (1 - \epsilon_1) |A| + \frac{c(c+1)}{2} \cdot \epsilon_2 |F|, \end{aligned}$$

where the first inequality follows from Claim 11, the second inequality follows from our assumption on  $\alpha_m$  and  $|S_m|$ , and the last equality follows from (8).

Rearranging gives that  $|A| < \frac{\epsilon_2(c+1)}{2\epsilon_1} \cdot c|F| = \frac{\delta(t+1)-1}{2t} \cdot c|F|$ , contradicting (3). ◁

Next, we will show that by flipping all the variables in  $S_m$ , where  $m$  satisfies the conclusion of Claim 12, one can reduce the size of the set of corrupt variables by a  $(1 - \epsilon_3)$ -fraction, thereby proving Lemma 10.

**Proof of Lemma 10.** Let  $m \in [c]$  satisfy the conclusion of Claim 12. Combining the two inequalities (6) and (7), one can infer that

$$\frac{|A|}{|A| + |B|} = \frac{1}{2} + \frac{|A| - |B|}{2(|A| + |B|)} \geq \frac{1}{2} + \frac{\epsilon_0 \delta^2 c |F|}{2c|F|} = \frac{1}{2} + \frac{\epsilon_0 \delta^2}{2}. \quad (12)$$

Therefore, it follows by (12) that

$$\alpha_m \geq (1 - \epsilon_1) \frac{|A|}{|A| + |B|} \geq (1 - \epsilon_1) \left( \frac{1}{2} + \frac{\epsilon_0 \delta^2}{2} \right). \quad (13)$$

## 61:14 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

It follows by (9) that

$$\begin{aligned} |F'| &= (|F| - |S_m \cap F|) + |S_m \cap Z| = |F| - (2\alpha_m - 1)|S_m| \\ &\leq |F| - \left(2(1 - \epsilon_1) \left(\frac{1}{2} + \frac{\epsilon_0 \delta^2}{2}\right) - 1\right) |S_m| = |F| - (\epsilon_3/\epsilon_2)|S_m| \leq |F| - \epsilon_3|F|, \end{aligned}$$

as needed, where the second equality follows by (10), the first inequality follows by (13), the last equality follows by the definition of  $\epsilon_3$  and the last inequality follows by Claim 12. ◀

We will conclude by the following inequality, which shows that for an arbitrary  $m \in [c]$ , flipping  $S_m$  would not significantly increase the number of corrupt variables.

▷ **Claim 13.** For arbitrary  $x \in \mathbb{F}_2^n$  and  $m \in [c]$ , let  $x' := \text{EasyFlip}(x, m)$ . Let  $F$  and  $F'$  be the sets of corrupt variables of  $x$  and  $x'$ , respectively. Then  $|F'| \leq (1 + \frac{c}{d_0 - t})|F|$ .

*Proof.* Since the constraints in  $A$  always compute the correct codewords in  $C_0$ , they always send correct flips to their neighbors in  $L$ . Therefore, the wrong flips must be sent by  $B$ . Therefore, in the worst case (i.e., assuming that  $A = \emptyset$ ), we have that

$$|F'| \leq |F| + |B| \leq \left(1 + \frac{c}{d_0 - t}\right) |F|,$$

where the second inequality follows from (5). ◀

### 3.3 Running EasyFlip iteratively for a constant number of times – DeepFlip

In this subsection, we will present and analyze DeepFlip (see Algorithm 3 below), which is designed by running EasyFlip iteratively for  $s$  times for a particular choice of  $(m_1, \dots, m_s) \in [c]^s$ . By iteratively we mean a sequence of operations  $x^0 := x, x^1 := \text{EasyFlip}(x^0, m_1), \dots, x^s = \text{EasyFlip}(x^{s-1}, m_s)$ .

■ **Algorithm 3** Running EasyFlip iteratively for a particular choice  $(m_1, \dots, m_s) \in [c]^s$  – DeepFlip.

---

**Input:**  $G, C_0, x \in \mathbb{F}_2^n$ , and  $(m_1, \dots, m_s) \in [c]^s$   
**Output:**  $x^s \in \mathbb{F}_2^n$  or  $\perp$

- 1 Set  $k = 1$  and  $x^0 = x$ ;
- 2 **for**  $1 \leq k \leq s$  **do**
- 3      $x^k \leftarrow \text{EasyFlip}(x^{k-1}, m_k)$ ;
- 4      $U^k \leftarrow \{u \in R : x_{N(u)}^k \notin C_0\}$ ;
- 5     **if**  $|U^k| > (1 - \epsilon_3)^k \cdot c\gamma n$  **then**
- 6         **return**  $\perp$
- 7     **else**
- 8          $k \leftarrow k + 1$
- 9     **end**
- 10 **end**
- 11 **return**  $x^s$

---

Our goal is to show that as long as the number of corrupt variables in  $x$  is not too large, by running EasyFlip iteratively for a large enough (but still constant) number of times, there exists a vector  $(m_1, \dots, m_s) \in [c]^s$  such that the number of corrupt variables in the final

output  $x^s$  is at most a  $(1 - \epsilon_3)$ -fraction of the number of corrupt variables in the initial input  $x$ . Most importantly, later we will show that such a vector  $(m_1, \dots, m_s)$  can be found *explicitly* and *efficiently*.

The above assertion will be made precise by the following lemma.

► **Lemma 14.** *Let  $x$  be the input vector of DeepFlip and let  $F$  be the set of corrupt variables of  $x$ . If  $|F| \leq \gamma n$ , then for every  $s \geq s_0$  there exists a nonempty subset  $M \subseteq [c]^s$  such that the following holds for every  $(m_1, \dots, m_s) \in M$ . Let  $x^s := \text{DeepFlip}(x, (m_1, \dots, m_s))$  be the output vector of DeepFlip and  $F^s$  be the set of corrupt variables of  $x^s$ . Then  $|F^s| \leq (1 - \epsilon_3)|F|$ .*

### 3.3.1 Proof of Lemma 14

Given  $(m_1, \dots, m_s) \in [c]^s$  and  $x^0 := x$ , for each  $k \in [s]$ , let  $x^k := \text{EasyFlip}(x^{k-1}, m_k)$ . With this notation,  $x^s = \text{EasyFlip}(x^{s-1}, m_s) = \text{DeepFlip}(x, (m_1, \dots, m_s))$ , is exactly the output vector of DeepFlip. Let  $F$  be the set of corrupt variables in  $x$  and  $U$  be the set of unsatisfied constraints with respect to  $x$ . Sometimes, we will also use  $F^0 := F$  and  $U^0 := U$ . For  $k \in [s]$ , define  $F^k$  and  $U^k$  similarly with  $x$  replaced by  $x^k$ . Then

$$N_{\leq d_0-1}(F) \subseteq U \subseteq N(F), \quad (14)$$

where the first inclusion holds since  $d(C_0) = d_0$ .

The following lemma can be viewed as an “idealized” version of Lemma 14.

► **Lemma 15.** *With the above notation, the following holds. If  $|F| \leq \alpha n$ , then there exists a vector  $(m_1, \dots, m_s) \in [c]^s$  such that*

- (i)  $|F^s| \leq (1 - \epsilon_3)^s |F|$ ;
- (ii) for each  $k \in [s]$ ,  $|U^k| \leq (1 - \epsilon_3)^k \cdot c|F|$ ;
- (iii)  $|U^s| \leq (1 - \epsilon_3)^s \cdot \frac{d_0-1}{\delta d_0-1} \cdot |U|$ .

**Proof.** As  $|F| \leq \alpha n$ , by Lemma 10, there exists  $m_1 \in [c]$  such that  $x^1 = \text{EasyFlip}(x, m_1)$  satisfies  $|F^1| \leq (1 - \epsilon_3)|F| \leq \alpha n$ . Continuing this process, it follows by Lemma 10 that for each  $k \in [s]$ , there exists  $m_k \in [c]$  such that  $x^k = \text{EasyFlip}(x^{k-1}, m_k)$  satisfies

$$|F^k| \leq (1 - \epsilon_3)|F^{k-1}| \leq (1 - \epsilon_3)^k |F| \leq \alpha n. \quad (15)$$

Such a vector  $(m_1, \dots, m_s) \in [c]^s$  clearly satisfies property (i).

To prove (ii), note that it follows by (14) and (15) that for each  $k \in [s]$ ,

$$|U^k| \leq |N(F^k)| \leq c|F^k| \leq (1 - \epsilon_3)^k \cdot c|F|.$$

To prove (iii), as  $|F| \leq \alpha n$ , applying Proposition 7 in concert with (14) gives that  $\frac{\delta d_0-1}{d_0-1} \cdot c|F| \leq |N_{\leq d_0-1}(F)| \leq |U|$ . Combining the equation above and (i) gives that  $|U^s| \leq c|F^s| \leq (1 - \epsilon_3)^s \cdot c|F| \leq (1 - \epsilon_3)^s \cdot \frac{d_0-1}{\delta d_0-1} \cdot |U|$ , completing the proof of (iii). ◀

Lemma 15 (i) indicates that there exists an “ideal” choice, say  $(m_1^*, \dots, m_s^*) \in [c]^s$ , such that if  $|F| \leq \alpha n$ , then after the execution of EasyFlip iteratively for  $s$  times (directed by  $(m_1^*, \dots, m_s^*)$ ), the number of corrupt variables in the final output  $x^s$  is at most a  $(1 - \epsilon_3)^s$ -fraction of the number of corrupt variables in the initial input  $x^0 = x$ .

Unfortunately, in general, there is no way to compute the number of corrupt variables in the input and output of each execution of EasyFlip. From this perspective, there is no easy way to *explicitly* find the ideal  $(m_1^*, \dots, m_s^*) \in [c]^s$ . However, Lemma 15 (iii), which is a consequence of Lemma 15 (i), essentially shows that if the number of corrupt variables

## 61:16 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

reduces *dramatically*, then the number of unsatisfied constraints also reduces *significantly* - fortunately, it is clear that this quantity can be computed in linear time! The analysis of our deterministic decoding algorithm relies heavily on this observation.

The above discussion motivates the following definition.

► **Definition 16.** *Given the input vector  $x$  of DeepFlip, let  $M$  be the set consisting of all vectors  $(m_1, \dots, m_s) \in [c]^s$  which satisfy the following two properties:*

- (a) *for each  $k \in [s]$ ,  $|U^k| \leq (1 - \epsilon_3)^k \cdot c\gamma n$ ;*
- (b)  *$|U^s| \leq \epsilon_4 |U|$ , where  $\epsilon_4 = \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1 - \epsilon_3)$ .*

The following result is an easy consequence of Lemma 15.

▷ **Claim 17.** *If  $|F| \leq \gamma n$  and  $s \geq s_0$ , then  $M \neq \emptyset$ .*

*Proof.* Since  $|F| \leq \gamma n < \alpha n$ , there exists a vector  $(m_1, \dots, m_s) \in [c]^s$  that satisfies Lemma 15. By substituting  $|F| \leq \gamma n$  into Lemma 15 (ii), it is easy to see that such a vector also satisfies Definition 16 (a). Moreover, by substituting  $s \geq s_0 = \left\lceil \log_{1-\epsilon_3} \left( \epsilon_4 \frac{\delta d_0 - 1}{d_0 - 1} \right) \right\rceil$  into Lemma 15 (iii), it is not hard to see that Definition 16 (b) also holds. Therefore,  $M \neq \emptyset$ , as needed. ◁

As briefly mentioned above, in general one cannot explicitly find the ideal  $(m_1^*, \dots, m_s^*) \in [c]^s$  which dramatically reduces the number of corruptions. Instead, under a stronger condition  $|F| \leq \gamma n$  (recall that Lemma 15 assumes  $|F| \leq \alpha n$ ), Lemma 14 shows that for every  $(m_1, \dots, m_s) \in M$ ,  $x^s = \text{DeepFlip}(x, (m_1, \dots, m_s))$  reduces the number of corrupt variables of  $x$  by a  $(1 - \epsilon_3)$ -fraction, which makes every member of  $M$  an *acceptable* (which may be not ideal) choice for DeepFlip.

Now we are ready to present the proof of Lemma 14.

**Proof of Lemma 14.** First of all, we would like to show that Lemma 14 is well defined, namely, for every  $|F| \leq \gamma n$  and  $(m_1, \dots, m_s) \in M$ ,  $\text{DeepFlip}(x, (m_1, \dots, m_s))$  does not return  $\perp$ . Indeed, as  $(m_1, \dots, m_s) \in M$ , by Definition 16 (a) we have that for every  $1 \leq k \leq s$ ,  $|U^k| \leq (1 - \epsilon_3)^k \cdot c\gamma n$ , which implies that  $U^k$  always passes the test in step 5 of Algorithm 3. Therefore, under the assumption of Lemma 14, the output of DeepFlip is a vector  $x^s \in \mathbb{F}_2^n$ .

To prove the lemma, assume for the moment that  $|F^s| \leq \alpha n$ . Given the correctness of this assertion, applying Proposition 7 in concert with (14) gives that

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^s| \leq |N_{\leq d_0 - 1}(F^s)| \leq |U^s|.$$

Moreover, by combining the above equation and Definition 16 (b), we have

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^s| \leq |U^s| \leq \epsilon_4 |U| \leq \frac{\delta d_0 - 1}{d_0 - 1} \cdot (1 - \epsilon_3) \cdot c|F|,$$

which implies that  $|F^s| \leq (1 - \epsilon_3)|F|$ , as needed.

Therefore, it remains to show that  $|F^s| \leq \alpha n$ . We will prove by induction that for each  $0 \leq k \leq s$ ,  $|F^k| \leq \frac{\alpha n}{1 + c/(d_0 - t)} \leq \alpha n$ . For the base case  $k = 0$ , it follows by assumption that  $|F^0| \leq \gamma n < \frac{\alpha n}{1 + c/(d_0 - t)}$  as  $d_0 \geq 3$ ,  $\delta d_0 > 3$  and  $t = \lfloor \frac{1}{\delta} \rfloor$ . Suppose that for some  $k \in [s]$  we have  $|F^{k-1}| \leq \frac{\alpha n}{1 + c/(d_0 - t)}$ . Since  $x^k = \text{EasyFlip}(x^{k-1}, m_k)$ , it follows by Claim 13 that  $|F^k| \leq (1 + \frac{c}{d_0 - t})|F^{k-1}| \leq \alpha n$ . Therefore, we have

$$\frac{\delta d_0 - 1}{d_0 - 1} \cdot c|F^k| \leq |N_{\leq d_0 - 1}(F^k)| \leq |U^k| \leq (1 - \epsilon_3)^k \cdot c\gamma n \leq c\gamma n,$$

where the first inequality follows from Proposition 7, the second inequality follows from (14), and the third inequality follows from Definition 16 (a). The last equation implies that  $|F^k| \leq \frac{d_0-1}{\delta d_0-1} \gamma n < \frac{d_0}{2} \gamma n \leq \frac{\alpha n}{1+c/(d_0-t)}$ , as needed, where the second inequality follows from the assumption  $\delta d_0 > 3$  and the last inequality follows from the definition of  $\gamma$  in Theorem 3,  $\delta d_0 > 3$  and  $t = \lfloor \frac{1}{\delta} \rfloor$ . The proof of the lemma is thus completed.  $\blacktriangleleft$

### 3.4 Running DeepFlip thoroughly until significantly reducing the number of unsatisfied constraints – HardSearch

In this subsection, we describe and analyze HardSearch (see Algorithm 4 below). Given an input vector  $x \in \mathbb{F}_2^n$  with at most  $\gamma n$  corruptions, HardSearch runs DeepFlip( $x, (m_1, \dots, m_s)$ ) over all choices of  $(m_1, \dots, m_s) \in [c]^s$  until it finds one, say  $(m'_1, \dots, m'_s)$ , such that the number of unsatisfied constraints with respect to DeepFlip( $x, (m'_1, \dots, m'_s)$ ) is at most an  $\epsilon_4$ -fraction of the number of unsatisfied constraints with respect to  $x$ . Then Lemma 8 shows that the number of corruptions in  $x'$  is at most a  $(1 - \epsilon_3)$ -fraction of the number of corruptions in  $x$ . Therefore, running HardSearch iteratively for  $\ell$  rounds gives us a  $(1 - \epsilon_3)^\ell$ -reduction on the number of corruptions.

■ **Algorithm 4** Running DeepFlip over all  $(m_1, \dots, m_s) \in [c]^s$  until finding an “acceptable” one – HardSearch.

---

**Input:**  $G, C_0, x \in \mathbb{F}_2^n$ , and  $s = s_0$   
**Output:**  $x' \in \mathbb{F}_2^n$

- 1  $U \leftarrow \{u \in R : x_{N(u)} \notin C_0\};$
- 2 **for** every  $(m_1, \dots, m_s) \in [c]^s$  **do**
- 3      $x' \leftarrow \text{DeepFlip}(x, (m_1, \dots, m_s));$
- 4     **if**  $x' \neq \perp$  **then**
- 5          $U' \leftarrow \{u \in R : x'_{N(u)} \notin C_0\};$
- 6         **if**  $|U'| \leq \epsilon_4 |U|$  **then**
- 7             return  $x'$
- 8         **end**
- 9     **end**
- 10 **end**

---

#### 3.4.1 Proof of Lemma 8

To prove (i), let  $M$  be the set of vectors in  $[c]^s$  which satisfy the two conditions in Definition 16 with respect to  $F$  and  $s$ , where  $|F| \leq \gamma n$  and  $s = s_0$ . By our choices of  $F$  and  $s$ , it follows by Claim 17 that  $M \neq \emptyset$ . By Lemma 14, as long as HardSearch finds a vector  $(m_1, \dots, m_s) \in M$ , it would output a vector  $x' = \text{DeepFlip}(x, (m_1, \dots, m_s))$  such that  $|U'| \leq \epsilon_4 |U|^4$  and  $|F'| \leq (1 - \epsilon_3) |F|$ , as needed.

It remains to prove (ii), which is an easy consequence of (i). Let  $F^i$  be the set of corruptions in  $x^i$  for all  $0 \leq i \leq \ell$ . Then by (i) for every  $0 \leq i \leq \ell - 1$ , we have either  $x^i \in T(G, C_0)$  (if  $|U^i| = 0$ ) or  $|F^{i+1}| \leq (1 - \epsilon_3) |F^i|$  (if  $|U^i| \neq 0$ ). Therefore, after at most  $\ell = \left\lceil \log_{1-\epsilon_3} \left( \left\lfloor \frac{d_0-1}{2} \right\rfloor \frac{1}{\gamma n} \right) \right\rceil$  iterative executions of HardSearch, the number of corrupt variables is at most  $(1 - \epsilon_3)^\ell \gamma n \leq \left\lfloor \frac{d_0-1}{2} \right\rfloor \frac{1}{\gamma n} \cdot \gamma n = \left\lfloor \frac{d_0-1}{2} \right\rfloor$ , as needed.

<sup>4</sup> This holds since  $(m_1, \dots, m_s) \in M$  satisfies Definition 16 (b).

#### 4 Randomized decoding: Reduce large corruptions to a moderate size

In this section, we present our randomized decoding for Tanner codes which can correct more errors. The general strategy is as follows. First, we use a voting process to derive a set  $S := \cup_{m \in [c]} S_m$  of candidate variables to flip. More precisely, each constraint  $u \in R$  that satisfies  $1 \leq d_H(\text{Decode}(x_{N(u)}), x_{N(u)}) \leq t$  sends exactly one flip to an arbitrary variable  $v \in N(u)$  with  $\text{Decode}(x_{N(u)})_v \neq x_v$ . We then design a special sampling process to pick a large fraction of variables from  $S$  and flip them. This process can, with high probability, reduce the number of corrupted variables by a positive fraction. We repeat the above random process until the number of corrupted variables drops below  $\gamma n$ , in which case our deterministic decoding MainDecode in Algorithm 1 can work correctly, or we run out of time and stop. Finally, we use MainDecode to get the codeword.

Let  $\gamma$  be the relative decoding radius of Theorem Theorem 3. The exact randomized decoding is given as Algorithm 5, which yields the following result.

##### Algorithm 5 Randomized Decoding.

---

**Input:**  $x \in \mathbb{F}_2^n$  with at most  $\alpha$  fraction errors  
**Output:** a codeword in  $T(G, C_0)$  or  $\perp$

- 1 Set  $t = \lfloor \frac{1}{\delta} \rfloor$ ;
- 2 **for**  $\ell = 1, \dots, \left\lceil \frac{\log \frac{\gamma}{\alpha}}{\log \left(1 - \frac{3\epsilon(\delta(t+1)-1)}{4t}\right)} \right\rceil$  **do**
- 3     **for every**  $u \in R$  **do**
- 4          $\omega \leftarrow \text{Decode}(x_{N(u)})$ ;
- 5         **if**  $1 \leq d_H(\omega, x_{N(u)}) \leq t$  **then**
- 6             send a “flip” message to the vertex  $v \in N(u)$  with the smallest index such  
            that  $\omega_v \neq x_v$
- 7         **end**
- 8     **end**
- 9      $\forall m \in [c], S_m \leftarrow \{v \in L : v \text{ receives } m \text{ “flip” messages}\}$ ;
- 10      $S \leftarrow \bigcup_m S_m$ ;
- 11     Randomly pick  $P \subseteq S$ : for every  $m \in [c]$ , for each variable in  $S_m$ , pick it with  
        probability  $\frac{m}{2c}$ , using independent randomness;
- 12     Flip all bits in  $P$ ;
- 13      $U \leftarrow \{u \in R : x_{N(u)} \notin C_0\}$ ;
- 14     **if**  $|U| \leq \left(\delta - \frac{1}{d_0}\right) c\gamma n$  **then**
- 15         return MainDecode( $x$ )
- 16     **end**
- 17 **end**
- 18 return  $\perp$

---

The following two lemmas demonstrate the correctness and linear running time of Algorithm 5, respectively<sup>5</sup>.

► **Lemma 18.** *If the input has distance at most  $\alpha n$  from a codeword, then with probability  $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$ , Algorithm 5 outputs the correct codeword.*

<sup>5</sup> We only prove Lemma 18. The proof of Lemma 19 can be found in the full version of this paper [10]

► **Lemma 19.** *If the input has distance at most  $\alpha n$  from a codeword, then Algorithm 5 runs in linear time.*

Assuming the correctness of the above lemmas, we can prove Theorem 6 as follows.

**Proof of Theorem 6.** If the input word has at most  $\alpha n$  errors, then by Lemma 18, with probability  $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$ , the decoding outputs the correct codeword. Furthermore, the running time is linear by Lemma 19. ◀

## 4.1 Proof of Lemma 18

Recall that we defined  $A$  as the set of constraints  $u \in R$  that sends a flip and  $\text{Decode}(x_{N(u)})$  computes the correct codeword in  $C_0$  (see (2)), and  $B$  to be the set of constraints  $u \in R$  that sends a flip and  $\text{Decode}(x_{N(u)})$  computes an incorrect codeword in  $C_0$  (see (4)). Also, recall we let  $\alpha_m$  denote the fraction of corrupt variables in  $S_m$  (see (10)) and we let  $\beta_m$  denote the fraction of flips sent from  $A$  to  $S_m$  among all flips received by  $S_m$  (see (11)). Now we let  $M := A \cup B$ .

First, we bound the size of  $P$  in an arbitrary iteration.

▷ **Claim 20.** For every constant  $\epsilon > 0$ , with probability  $\geq 1 - \exp\{-\Theta_{c,\epsilon}(|M|)\}$ , the size of  $P$  is in  $\left[(1 - \epsilon)\frac{|M|}{2c}, (1 + \epsilon)\frac{|M|}{2c}\right]$ .

*Proof.* In Algorithm 5, for every  $m \in [c]$ , each variable in  $S_i$  is picked independently with probability  $\frac{m}{2c}$ . For each  $v \in S$ , let  $X_v$  be the indicator random variable of the event that the variable  $v$  is picked. So for every  $v \in S_m$ ,  $\Pr[X_v = 1] = \frac{m}{2c}$ . Let  $X = \sum_{v \in S} X_v$ . It is easy to see that  $X = |P|$ . By the linearity of expectation, we have that

$$\mathbb{E}X = \sum_{v \in S} \mathbb{E}X_v = \sum_{m \in [c]} \frac{m}{2c} |S_m| = \frac{|M|}{2c}.$$

By Hoeffding's inequality,

$$\Pr \left[ X \in \left[ (1 - \epsilon)\frac{|M|}{2c}, (1 + \epsilon)\frac{|M|}{2c} \right] \right] \geq 1 - 2 \exp \left\{ -\frac{2 \left( \epsilon \frac{|M|}{2c} \right)^2}{|S|} \right\} \geq 1 - 2 \exp \left\{ -\frac{\epsilon^2 |M|}{2c^2} \right\},$$

where the second inequality follows from the fact that  $|S| \leq |M|$ . ◀

Next, we show that  $P$  contains significantly more corrupted variables than correct variables.

▷ **Claim 21.** There exists a constant  $\epsilon$  such that with probability  $\geq 1 - \exp\{-\Theta_{c,\epsilon}(|M|)\}$ , the number of corrupted variables in  $P$  is at least  $(1/2 + \epsilon)\frac{|M|}{2c}$ .

*Proof of Claim 21.* For every  $v \in S$ , let  $Y_v$  be the indicator random variable of the event that  $X_v = 1$  and  $v \in F$ . Let  $Y = \sum_{v \in S} Y_v$ . By definition,  $Y = P \cap F$ . Note that for every  $v \notin S \cap F$ ,  $\Pr[Y_v = 1] = 0$ . By the linearity of expectation, we have that

$$\mathbb{E}Y = \sum_{v \in S} \mathbb{E}Y_v = \sum_{m \in [c]} \sum_{v \in S_m} \mathbb{E}Y_v = \sum_{m \in [c]} \sum_{v \in S_m \cap F} \frac{m}{2c} = \sum_{m \in [c]} \frac{m}{2c} \alpha_m |S_m| \geq \sum_{m \in [c]} \frac{m}{2c} \beta_m |S_m|, \quad (16)$$

where the inequality follows from Claim 11.

## 61:20 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

By the definition of  $\beta_m$ ,  $m\beta_m|S_m| = |\{\text{The number of "flips" sent from } A \text{ to } S_m\}|$ . Hence, one can infer that

$$\begin{aligned} \text{E}Y &\geq \sum_{m \in [c]} \frac{|\{\text{The number of "flips" sent from } A \text{ to } S_m\}|}{2c} \\ &= \frac{|\{\text{The number of "flips" sent from } A \text{ to } S\}|}{2c} = \frac{|A|}{2c}, \end{aligned}$$

where the last equality is due to that each constraint in  $R$  can only send at most 1 message.

Set  $\epsilon = \frac{\epsilon_0 \delta^2}{4} > 0$ . It follows by (12) that  $|A| \geq \left(\frac{1}{2} + \frac{\epsilon_0 \delta^2}{2}\right) |M| = \left(\frac{1}{2} + 2\epsilon\right) |M|$ . Thus, we can infer that  $\text{E}Y \geq \frac{|A|}{2c} \geq \left(\frac{1}{2} + 2\epsilon\right) \frac{|M|}{2c}$ .

By Hoeffding's inequality,

$$\Pr \left[ Y \leq \left(\frac{1}{2} + \epsilon\right) \frac{|M|}{2c} \right] \geq 1 - \exp \left\{ -\frac{2 \left(\epsilon \frac{|M|}{2c}\right)^2}{|S|} \right\} \geq 1 - \exp \left\{ -\frac{\epsilon^2 |M|}{2c^2} \right\},$$

where the second inequality follows from that  $|S| \leq |M|$ .  $\triangleleft$

The following claim shows that as long as the number of unsatisfied constraints is small enough, we can ensure that the number of corrupt variables is at most  $\gamma n$ . Hence, we can handle the matter with Algorithm 1.

$\triangleright$  **Claim 22.** If  $|U| \leq \left(\delta - \frac{1}{d_0}\right) c\gamma n$  and  $|F| \leq \alpha n$ , then  $|F| \leq \gamma n$ .

*Proof.* Suppose that  $\gamma n < |F| \leq \alpha n$ . By Proposition 7, we have that

$$|U| \geq \frac{\delta d_0 - 1}{d_0 - 1} c|F| > \left(\delta - \frac{1}{d_0}\right) c\gamma n,$$

which is a contradiction.  $\triangleleft$

Now, we can give the proofs of Lemma 18 and Lemma 19, respectively, as follows.

**Proof of Lemma 18.** In each iteration, consider the case that the number of errors  $|F|$  is at most  $\alpha n$ . If  $|U| \leq \left(\delta - \frac{1}{d_0}\right) c\gamma n$ , then by Claim 22,  $|F| \leq \gamma n$ . Therefore, it follows by Theorem 3 that when  $\delta d_0 > 3$ , all errors can be corrected. Otherwise, we claim that the number of corrupt variables can be decreased by a constant fraction in this iteration.

Recall that  $M = A \cup B \subseteq U$ . It follows by (3) that

$$|M| = |A| + |B| \geq \frac{\delta(t+1) - 1}{t} c|F|. \quad (17)$$

Note that by Claim 21, with probability  $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$ , the number of corruptions in  $P$  is at least  $(1/2 + \epsilon) \frac{|M|}{2c}$  where  $\epsilon > 0$  is a constant. Also note that by Claim 20, with probability  $1 - \exp\{-\Theta_{c,\delta,d_0}(n)\}$ , the size of  $P$  is in  $\left[(1 - \epsilon/2) \frac{|M|}{2c}, (1 + \epsilon/2) \frac{|M|}{2c}\right]$ . When both of these events occur, by flipping all variables in  $\hat{P}$ , the number of corruptions is reduced by at least  $\frac{3\epsilon|M|}{4c}$ . It follows by (17) that  $\frac{3\epsilon|M|}{4c} \geq \frac{3\epsilon(\delta(t+1)-1)}{4t} |F|$ . This shows the number of corrupt variables indeed is decreased by a constant fraction in this iteration.

As a result, after at most  $\frac{\log \frac{\gamma}{\alpha}}{\log\left(1 - \frac{3\epsilon(\delta(t+1)-1)}{4t}\right)}$  iterations, the number of corruptions is at most  $\gamma n$ . Then the decoding can call Algorithm 1 to correct all errors.  $\blacktriangleleft$



## References

- 1 Noga Alon and Michael Capalbo. Explicit unique-neighbor expanders. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 73–79. IEEE, 2002.
- 2 Sanjeev Arora, Constantinos Daskalakis, and David Steurer. Message-passing algorithms and improved lp decoding. *IEEE Trans. Inf. Theory*, 58(12):7260–7271, 2012. doi:10.1109/TIT.2012.2208584.
- 3 Ron Asherov and Irit Dinur. Bipartite unique neighbour expanders via ramanujan graphs. *Entropy*, 26(4):348, 2024.
- 4 Oren Becker. Symmetric unique neighbor expanders and good ldpc codes. *Discrete Applied Mathematics*, 211:211–216, 2016.
- 5 Avraham Ben-Aroya and Amnon Ta-Shma. A combinatorial construction of almost-ramanujan graphs using the zig-zag product. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 325–334, 2008.
- 6 Nikolas P. Breuckmann and Jens Niklas Eberhardt. Balanced product quantum codes. *IEEE Trans. Inf. Theory*, 67(10):6653–6674, 2021. doi:10.1109/TIT.2021.3097347.
- 7 Nikolas P Breuckmann and Jens Niklas Eberhardt. Quantum low-density parity-check codes. *PRX Quantum*, 2(4):040101, 2021.
- 8 Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 659–668, 2002.
- 9 Xue Chen, Kuan Cheng, Xin Li, and Minghui Ouyang. Improved decoding of expander codes. *IEEE Trans. Inf. Theory*, 69(6):3574–3589, 2023. doi:10.1109/TIT.2023.3239163.
- 10 Kuan Cheng, Minghui Ouyang, Chong Shangguan, and Yuanting Shen. Improved decoding of expander codes: fundamental trade-off between expansion ratio and minimum distance of inner code. *arXiv preprint*, 2023. arXiv:2312.16087.
- 11 Shashi Kiran Chilappagari, Dung Viet Nguyen, Bane Vasic, and Michael W. Marcellin. On trapping sets and guaranteed error correction capability of LDPC codes and GLDPC codes. *IEEE Trans. Inf. Theory*, 56(4):1600–1611, 2010. doi:10.1109/TIT.2010.2040962.
- 12 Sae-Young Chung, G David Forney, Thomas J Richardson, and Rüdiger Urbanke. On the design of low-density parity-check codes within 0.0045 db of the shannon limit. *IEEE Communications letters*, 5(2):58–60, 2001.
- 13 Itay Cohen, Roy Roth, and Amnon Ta-Shma. Hdx condensers. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1649–1664. IEEE, 2023.
- 14 Alexandros G. Dimakis, Roxana Smarandache, and Pascal O. Vontobel. Ldpc codes for compressed sensing. *IEEE Trans. Inf. Theory*, 58(5):3093–3114, 2012. doi:10.1109/TIT.2011.2181819.
- 15 Irit Dinur, Min-Hsiu Hsieh, Ting-Chun Lin, and Thomas Vidick. Good quantum LDPC codes with linear time decoders. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 905–918. ACM, 2023. doi:10.1145/3564246.3585101.
- 16 Michael Dowling and Shuhong Gao. Fast decoding of expander codes. *IEEE Trans. Inf. Theory*, 64(2):972–978, 2018. doi:10.1109/TIT.2017.2726064.
- 17 Shai Evra, Tali Kaufman, and Gilles Zémor. Decodable quantum LDPC codes beyond the  $\sqrt{n}$  distance barrier using high dimensional expanders. *CoRR*, abs/2004.07935, 2020. arXiv:2004.07935.
- 18 Jon Feldman, Tal Malkin, Rocco A. Servedio, Cliff Stein, and Martin J. Wainwright. Lp decoding corrects a constant fraction of errors. *IEEE Trans. Inf. Theory*, 53(1):82–89, 2007. doi:10.1109/TIT.2006.887523.
- 19 Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

## 61:22 When Can an Expander Code Correct $\Omega(n)$ Errors in $O(n)$ Time?

- 20 Louis Golowich. New explicit constant-degree lossless expanders. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4963–4971. SIAM, 2024.
- 21 Shouzhen Gu, Christopher A. Pattison, and Eugene Tang. An efficient decoder for a linear distance quantum LDPC code. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 919–932. ACM, 2023. doi:10.1145/3564246.3585169.
- 22 Venkatesan Guruswami. Iterative decoding of low-density parity check codes (a survey). *arXiv preprint cs/0610022*, 2006.
- 23 Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 812–821. ACM, 2002. doi:10.1145/509907.510023.
- 24 Matthew B. Hastings, Jeongwan Haah, and Ryan O’Donnell. Fiber bundle codes: breaking the  $n^{1/2}$  polylog( $n$ ) barrier for quantum LDPC codes. 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 1276–1288. ACM, 2021. doi:10.1145/3406325.3451005.
- 25 Jun-Ting Hsieh, Theo McKenzie, Sidhanth Mohanty, and Pedro Paredes. Explicit two-sided unique-neighbor expanders. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 788–799, 2024.
- 26 Tali Kaufman and Izhar Oppenheim. Construction of new local spectral high dimensional expanders. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 773–786, 2018.
- 27 Tali Kaufman and Ran J. Tessler. New cosystolic expanders from tensors imply explicit quantum LDPC codes with  $\Omega(\sqrt{n} \log^k n)$  distance. 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 1317–1329. ACM, 2021. doi:10.1145/3406325.3451029.
- 28 Swastik Kopparty, Noga Ron-Zewi, and Shubhangi Saraf. Simple constructions of unique neighbor expanders from error-correcting codes. *arXiv preprint*, 2023. arXiv:2310.19149.
- 29 Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. Quantum expander codes. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 810–824. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.55.
- 30 Anthony Leverrier and Gilles Zémor. Quantum tanner codes. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 872–883. IEEE, 2022. doi:10.1109/FOCS54457.2022.00117.
- 31 Anthony Leverrier and Gilles Zémor. Decoding quantum tanner codes. *IEEE Trans. Inf. Theory*, 69(8):5100–5115, 2023. doi:10.1109/TIT.2023.3267945.
- 32 Anthony Leverrier and Gilles Zémor. Efficient decoding up to a constant fraction of the code length for asymptotically good quantum codes. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1216–1244. SIAM, 2023. doi:10.1137/1.9781611977554.CH45.
- 33 Ting-Chun Lin and Min-Hsiu Hsieh. Good quantum ldpc codes with linear time decoder from lossless expanders. *arXiv preprint*, 2022. arXiv:2203.03581.
- 34 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- 35 Michael G Luby, Michael Mitzenmacher, Mohammad Amin Shokrollahi, and Daniel A Spielman. Efficient erasure correcting codes. *IEEE Trans. Inf. Theory*, 47(2):569–584, 2001.
- 36 GA Margulis. Explicit constructions of expanders (russian), *problemy peredaci informacii* 9 (1973), no. 4, 71–80. MR0484767, 1973.




- 37 Moshe Morgenstern. Existence and explicit constructions of  $q+1$  regular ramanujan graphs for every prime power  $q$ . *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994.
- 38 Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. Ldpc codes achieve list decoding capacity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 458–469, 2020. doi:10.1109/FOCS46700.2020.00050.
- 39 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. 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 375–388. ACM, 2022. doi:10.1145/3519935.3520017.
- 40 Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Trans. Inf. Theory*, 68(1):213–229, 2022. doi:10.1109/TIT.2021.3119384.
- 41 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 3–13. IEEE, 2000.
- 42 Thomas J Richardson, Mohammad Amin Shokrollahi, and Rüdiger L Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inf. Theory*, 47(2):619–637, 2001.
- 43 Thomas J Richardson and Rüdiger L Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory*, 47(2):599–618, 2001.
- 44 Noga Ron-Zewi, Mary Wootters, and Gilles Zémor. Linear-time erasure list-decoding of expander codes. *IEEE Trans. Inf. Theory*, 67(9):5827–5839, 2021. doi:10.1109/TIT.2021.3086805.
- 45 Ron M. Roth and Vitaly Skachek. Improved nearly-mds expander codes. *IEEE Trans. Inf. Theory*, 52(8):3650–3661, 2006. doi:10.1109/TIT.2006.878232.
- 46 Michael Sipser and Daniel A Spielman. Expander codes. *IEEE Trans. Inf. Theory*, 42(6):1710–1722, 1996.
- 47 Vitaly Skachek. Minimum distance bounds for expander codes. In *2008 Information Theory and Applications Workshop*, pages 366–370. IEEE, 2008.
- 48 DA Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996.
- 49 R Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inf. Theory*, 27(5):533–547, 1981.
- 50 Michael Viderman. Linear-time decoding of regular expander codes. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–25, 2013.
- 51 Michael Viderman. LP decoding of codes with expansion parameter above  $2/3$ . *Inf. Process. Lett.*, 113(7):225–228, 2013. doi:10.1016/J.IPL.2013.01.012.
- 52 Gilles Zémor. On expander codes. *IEEE Trans. Inf. Theory*, 47(2):835–837, 2001. doi:10.1109/18.910593.
- 53 Gillés Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.



# Coboundary and Cosystolic Expansion Without Dependence on Dimension or Degree

Yotam Dikstein   

Institute for Advanced Study, Princeton, NJ, USA

Irit Dinur   

Weizmann Institute of Science, Rehovot, Israel

---

## Abstract

We give new bounds on the cosystolic expansion constants of several families of high dimensional expanders, and the known coboundary expansion constants of order complexes of homogeneous geometric lattices, including the spherical building of  $SL_n(\mathbb{F}_q)$ . The improvement applies to the high dimensional expanders constructed by Lubotzky, Samuels and Vishne, and by Kaufman and Oppenheim.

Our new expansion constants do not depend on the degree of the complex nor on its dimension, nor on the group of coefficients. This implies improved bounds on Gromov’s topological overlap constant, and on Dinur and Meshulam’s cover stability, which may have applications for agreement testing.

In comparison, existing bounds decay exponentially with the ambient dimension (for spherical buildings) and in addition decay linearly with the degree (for all known bounded-degree high dimensional expanders). Our results are based on several new techniques:

- We develop a new “color-restriction” technique which enables proving dimension-free expansion by restricting a multi-partite complex to small random subsets of its color classes.
- We give a new “spectral” proof for Evra and Kaufman’s local-to-global theorem, deriving better bounds and getting rid of the dependence on the degree. This theorem bounds the cosystolic expansion of a complex using coboundary expansion and spectral expansion of the links.
- We derive absolute bounds on the coboundary expansion of the spherical building (and any order complex of a homogeneous geometric lattice) by constructing a novel family of very short cones.

**2012 ACM Subject Classification** Theory of computation → Expander graphs and randomness extractors

**Keywords and phrases** High Dimensional Expanders, HDX, Spectral Expansion, Coboundary Expansion, Cocycle Expansion, Cosystolic Expansion

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.62

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2304.01608> [10]

**Funding** Both authors were supported by Irit Dinur’s ERC grant 772839, and ISF grant 2073/21 when working on this project.

*Yotam Dikstein*: This material is based upon work supported by the National Science Foundation under Grant No. DMS-1926686.

**Acknowledgements** We are deeply grateful to Lewis Bowen for his important feedback on our paper, which greatly improved its clarity and readability.

## 1 Introduction

High dimensional expansion, which is a generalization of graph expansion to higher dimensional objects, is an active topic in recent years. The importance of graph expansion across many areas of computer science and mathematics, suggests that high dimensional expansion



© Yotam Dikstein and Irit Dinur;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 62; pp. 62:1–62:24



Leibniz International Proceedings in Informatics

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

may also come to have significant impact. So far we have seen several exciting applications including analysis of convergence of Markov chains [1], and constructions of locally testable codes and quantum LDPC codes [16, 55].

Several notions of expansion that are equivalent in graphs, such as convergence of random walks, spectral expansion, and combinatorial expansion, turn out to diverge into two main notions in higher dimensions.

The first is the notion of local link expansion which has to do with the expansion of the graph underlying each of the links of the complex; where a link is a sub-complex obtained by taking all faces that contain a fixed lower-dimensional face. This notion is qualitatively equivalent to convergence of random walks, it implies agreement testing, and it captures a spectral similarity between a (possibly sparse) high dimensional expander and the dense complete complex. It allows a spectral decomposition of functions on the faces of the complex in the style of Fourier analysis on the Boolean hypercube, see [13, 41, 31, 3, 25].

The second notion is coboundary and cosystolic expansion. Here we look at the complex not only as a combinatorial object but also as a sequence of linear maps, called coboundary maps, defined by the incidence relations of the complex. The  $i$ -th coboundary map  $\delta_i$  maps a function on the  $i$ -faces to a function on the  $i + 1$ -faces,  $C^0 \xrightarrow{\delta_0} C^1 \xrightarrow{\delta_1} \dots \xrightarrow{\delta_{d-1}} C^d$  where  $C^i = C^i(X, \mathbb{F}_2) = \{f : X(i) \rightarrow \mathbb{F}_2\}$  is the space of functions on  $i$  faces with coefficients in  $\mathbb{F}_2$  (we will consider general groups of coefficients, beyond  $\mathbb{F}_2$ ). These functions are called  $i$ -chains. The coboundary map  $\delta_i$  is defined in a very natural way: the value of  $\delta f(s)$  for any  $s \in X(i + 1)$  is the sum of  $f(t)$  for all  $s \supset t \in X(i)$  (the precise definition is in Section 2).

Coboundary (or cosystolic<sup>1</sup>) expansion captures how well the coboundary map tests its own kernel, in the sense of property testing. Given  $f \in C^i$  such that  $\delta f \approx 0$ , coboundary expansion guarantees existence of some  $g \in \ker \delta_i$  such that  $f \approx g$ . More precisely, a complex is a  $\beta$  coboundary (or cosystolic) expander if

$$wt(\delta f) \geq \beta \cdot \min_{g \in \ker \delta} \text{dist}(f, g)$$

where  $wt(\delta f)$  is the hamming weight of  $\delta f$ . We denote by  $h^i(X)$  the largest value of  $\beta$  that satisfies the above inequality for all  $f$ .

Whereas for  $i = 0$  coboundary expansion coincides with the combinatorial definition of edge expansion, for larger  $i$ , it may appear at first glance to be quite mysterious. However, this definition is far from being a merely syntactical generalization of the  $i = 0$  case and turns out to provide a rich connection between topological and cohomological concepts and between several important concepts in TCS, which we describe briefly below.

The study of coboundary and cosystolic expansion was initiated independently by Linial, Meshulam and Wallach [45], [51] in their study of connectivity of random complexes, and by Gromov [29] in his work on the topological overlapping property. Kaufman and Lubotzky [36] were the first to realize the connection between this definition and property testing. This point of view is important in the recent breakthroughs constructing locally testable codes and quantum LDPC codes [16, 55] (see also earlier works [23]).

Moreover, the coboundary maps come from a natural way to associate a (simplicial) complex to a constraint satisfaction problem. Attach a Boolean variable to each  $i$ -face, and view the  $(i + 1)$ -faces as parity constraints. The value that an assignment  $f : X(i) \rightarrow \mathbb{F}_2$

<sup>1</sup> The difference between coboundary and cosystolic expansion is just whether the cohomology is 0 or not (i.e. whether  $\ker \delta_{i+1} = \text{Im} \delta_i$ ). This distinction is not important for this exposition and the expansion inequality is the same in both cases.

gives on  $s \in X(i + 1)$  is  $\delta f(s)$ . This connection to CSPs has been harnessed towards showing that the CSPs derived from certain cosystolic expanders are hard to refute for resolution and for the sum of squares hierarchy, [17, 33].

In addition, cosystolic expansion of 1-chains (with non-abelian coefficients) of a complex has been connected to the stability of its topological covers [20]. Informally, a complex is cover-stable if slightly faulty simplicial covers are always “fixable” to valid simplicial covers. Perhaps surprisingly, this is related to agreement testing questions, particularly in the small 1% regime, which is a basic PCP primitive and part of the initial motivation for this work. We discuss agreement testing and its relation to coboundary expansion in more detail further below in this introduction.

In light of all of the above, we believe that cosystolic expansion is a fundamental notion that merits a deeper systematic study. Along with the aim of exploring its various implications, a more concrete research goal would be to give strong bounds, and ultimately nail down exactly, the correct expansion values for the most important and well-studied high dimensional expanders. We mention that to the best of our knowledge even for the simplest cases, such as expansion of  $k$ -chains in the  $n$ -simplex, exact expansion values are not yet completely determined.

In this work we provide new bounds for the coboundary expansion of the spherical building, and the cosystolic expansion of known bounded-degree high dimensional expanders including the complexes of [49, 48, 42].

Two of the most celebrated results in this area are the works of [35] and [22] showing that the bounded-degree families of Ramanujan complexes of [48] are cosystolic expanders. These works introduce an elegant local-to-global criterion, showing that if the links are coboundary expanders, and further assuming spectral expansion, then the entire complex is a cosystolic expander.

The estimates proven by [35, 22] for the coboundary expansion parameters are roughly  $h^k(X) \geq \min\left(\frac{1}{Q}, (d!)^{-O(2^k)}\right)$ . Here  $X$  is a  $d$  dimensional LSV complex and  $Q$  is the maximal degree of a vertex which is roughly equal to  $1/\lambda^{O(d^2)}$  in these complexes, where  $\lambda$  is the spectral bound on the expansion of the links. Subsequent works by Kaufman and Mass [37, 38, 39], improved this bound to

$$h^k(X) \geq \min\left(\frac{1}{Q}, (d!)^{-O(k)}\right). \tag{1}$$

We completely get rid of the dependence on the ambient dimension  $d$  and on the maximal degree  $Q$ , and prove

► **Theorem 1.** *For every integer  $d > 1$  and every small enough  $\lambda > 0$  let  $X$  be a  $d$ -dimensional LSV complex whose links are  $\lambda$ -one-sided expanders. For every group <sup>2</sup>  $\Gamma$ , every small enough  $\lambda > 0$  and every integer  $k < d - 1$ ,  $h^k(X, \Gamma) \geq \exp(-O(k^6 \log k))$ .*

Our bounds for  $h^k$  only depend on the dimension  $k$  of the chains, so for  $k = 1$  they are absolute constants. For larger  $k$  we still suffer an exponential decay. We do not know what the correct bound should be and whether dependence on  $k$  is at all necessary.

The case of  $k = 1$  is interesting even in complexes whose dimension is  $d \gg 1$ , because  $h^1$  controls the cover stability of the complex, as shown in [20]. Our bounds also immediately give an improvement for the topological overlap constants, when plugged into the Gromov machinery [30, 21, 22]. We elaborate on both of these applications later below.

---

<sup>2</sup> The theorem holds for every group  $\Gamma$  for which cohomology is defined, namely, abelian groups for  $k > 1$  and any group for  $k = 1$ .

The result is proven by enhancing the local-to-global criterion of [22], and introducing a variant of the local correction algorithm that makes local fixes only if they are sufficiently cost-effective. This is inspired by and resembles the algorithms in [22, 16, 55].

Our analysis is novel and departs from previous proofs: instead of relying on the so-called “fat machinery” of [22] (and its adaptations [37, 38]), our proof is 100% fat free and relies on the up/down averaging operators on *real-valued functions*. Our main argument is to show that, for a function  $h$  that is the indicator of the support of a (locally minimal)  $k$ -chain,

$$\|D \cdots Dh\|^2 \gtrsim \cdots \gtrsim \|DDh\|^2 \gtrsim \|Dh\|^2 \gtrsim \|h\|^2,$$

where  $D$  is the down averaging operator, and we write  $a \gtrsim b$  whenever  $a \geq \Omega(b)$ . From here we easily derive a lower bound on  $\|h\|^2$  showing that either the correction algorithm has found a nearby cocycle, or else the coboundary of our function was initially very large to begin with.

This method gives universal bounds on the cosystolic expansion of any complex whose links have both sufficient coboundary-expansion and sufficient local spectral expansion,

► **Theorem 2.** *Let  $\beta, \lambda > 0$  and let  $k > 0$  be an integer. Let  $X$  be a  $d$ -dimensional simplicial complex for  $d \geq k + 2$  and assume that  $X$  is a  $\lambda$ -one-sided local spectral expander. Let  $\Gamma$  be any group. Assume that for every non-empty  $r \in X$ ,  $X_r$  is a coboundary expander and that  $h^{k+1-|r|}(X_r, \Gamma) \geq \beta$ . Then*

$$h^k(X, \Gamma) \geq \frac{\beta^{k+1}}{(k+2)! \cdot 4} - e\lambda.$$

Here  $e \approx 2.71$  is Euler’s number.

Armed with an improved local-to-global connection, we derive Theorem 1 from Theorem 2 by further strengthening the coboundary expansion of the links of the LSV complexes, namely spherical buildings. The best previously known bound on coboundary expansion of  $k$ -cochains in spherical buildings is due to [30] and [47]. They proved a lower bound of  $\left(\binom{d+1}{k+1}(d+2)!\right)^{-1}$ . This decays exponentially with the ambient dimension  $d$ , and with the cochain level  $k$ . We remove the dependence on  $d$  by developing a new technique which we call “color-restriction”. The  $d$ -dimensional spherical buildings are colored, namely, they are  $d+1$ -partite. For a set of  $\ell$  colors  $F \subset [d+1]$ , the color restriction  $X^F$  is the complex induced on vertices whose color is contained in  $F$ . The restriction to the colors of  $F$  reduces the dimension of  $X$  from  $d$  to  $\ell - 1$ . We say that a color restriction  $X^F$  is a  $\beta$ -local coboundary expander, if  $X^F$  is a  $\beta$ -coboundary expander, and the same holds for the intersection of  $X^F$  with links (neighbourhoods) of faces whose color is disjoint from  $F$ . We show that if a typical color-restriction is a local coboundary expander, then the entire complex is a coboundary expander, and the expansion is independent of the dimension. Namely,

► **Theorem 3.** *Let  $k, \ell, d$  be integers so that  $k + 2 \leq \ell \leq d$  and let  $\beta, p \in (0, 1]$ . Let  $X$  be a  $(d + 1)$ -partite  $d$ -dimensional simplicial complex so that*

$$\mathbb{P}_{F \in \binom{[d+1]}{\ell}} [X^F \text{ is a } \beta\text{-locally coboundary expander}] \geq p.$$

Then  $h^k(X) \geq \frac{p\beta^{k+1}}{e(k+2)!}$ .



Finally, to prove that the spherical building satisfies the conditions of this theorem, we need to show that a typical random color-restriction is a good coboundary expander. For this we rely on the “cone machinery” developed by Gromov [30], Kozlov and Meshulam [44], and Kaufman and Oppenheim [42]. We construct in the full version of this paper [10], a novel family of short cones, thus proving the following.

► **Theorem 4.** *Let  $k \geq 0$ . There is an absolute constant  $\beta_k = \exp(-O(k^5 \log k)) \geq 0$  so that the following holds. Let  $X$  be the  $SL_n(\mathbb{F}_q)$ -spherical building for any integer  $n \geq k + 1$  and prime power  $q$ . Let  $\Gamma$  be any group. Then  $X$  is a coboundary expander with constant  $h^k(X, \Gamma) \geq \beta_k$ .*

In fact, we prove a more general version of this theorem, that holds for the order complex of any homogeneous geometric lattice, see the full version of this paper [10].

Most earlier works on cosystolic expansion focus on  $\mathbb{F}_2$  coefficients (see [37] and [20] for two exceptions). This is an important case especially in light of Gromov’s result connecting  $\mathbb{F}_2$ -expansion and topological overlap. However, expansion (of 1-chains) with respect to more general coefficients is necessary for results on topological covers and in turn for agreement testing. The theorems stated above show expansion of  $k$ -chains with respect to coefficients not only in  $\mathbb{F}_2$  but in general abelian groups  $\Gamma$ , and when  $k = 1$  also for non abelian groups  $\Gamma$ . In other words, the theorems hold for all groups of coefficients where the cohomology is defined.

Finally, we end with an upper bound. While most of our work is focused on lower bounds for coboundary and cosystolic expansion, we show in the full version of this paper [10] that families of dense simplicial complexes cannot have cosystolic expansion greater than  $1 + o(1)$ . This implies that high degree, in some weak sense, limits cosystolic expansion. It is interesting to compare this to a result of Kozlov and Meshulam that shows upper bounds on coboundary expansion of complexes with bounded degree [44].

### 1.1 Applications of cosystolic expansion

We describe two applications of cosystolic expansion for deriving topological properties of simplicial complexes.

#### Topological overlap

Cosystolic expansion was studied by [30] to give a combinatorial criterion for the topological overlapping property. Let  $f : X \rightarrow \mathbb{R}^k$  be continuous mapping (with respect to the natural topology on  $X$ ), i.e.  $f$  realizes  $X$  in  $\mathbb{R}^k$ . A point  $p \in \mathbb{R}^k$  is called  $c$ -heavily covered if

$$\mathbb{P}_{s \in X^{(k)}} [p \in f(s)] \geq c.$$

A well known result by [24] showed that for every affine map from the complete 2-dimensional complex to the plane, there exists a  $\frac{1}{27}$ -heavily covered point. Gromov’s greatly generalized this theorem to all *continuous* functions (instead of only affine functions), all dimensions  $k$  (instead of  $k = 2$ ) and complexes that are cosystolic expanders (instead of the complete complex), with  $c$  that depends on the dimension of the map  $k$ , as well as the cosystolic expansion constant. For a precise statement, see the full version of this paper [10].

The motivation for [22] was to show that there exists families of bounded degree simplicial complexes which have this property. They use [48] complexes and achieve a lower bound of  $c \geq \min(\frac{1}{Q}, (d!)^{-O(2^k)})$ , which comes from their bound on cosystolic expansion. This bound has been improved as a corollary of [39] to  $\min(\frac{1}{Q}, (d!)^{-O(k)})$ . Here again,  $d$  is the dimension of  $X$ , which may be much larger than  $k$ , and  $Q$  is the maximal degree of a vertex in  $X$ .

Plugging in our bounds into Gromov’s theorem gives the bound  $c \geq \exp(-O(k^7 \log k))$  for the topological overlapping property. This bound is free of the ambient dimension and of the degree.

### Cover stability

The second author and Meshulam studied a topological locally testable property called *cover stability* [20]. This property is equivalent to cosystolic expansion of 1-chains. A covering map between two simplicial complexes  $X, Y$  is a surjective  $t$ -to-1 simplicial map<sup>3</sup>  $\rho : Y(0) \rightarrow X(0)$  such that for every  $\tilde{u} \in Y(0)$  and  $\rho(\tilde{u}) = u \in X(0)$ , it holds that the links of  $\tilde{u}, u$  are isomorphic  $Y_{\tilde{u}} \cong X_u$ .

Graph covers (also known as lifts) have been quite useful in construction of expander graphs. Bilu and Linial showed that random covers of Ramanujan graphs are almost Ramanujan [6]. A celebrated result by [50] used these techniques to construct bipartite Ramanujan graphs of every degree. Recently, [9] showed that random covers could also be applied for constructing new simplicial complexes that are local spectral expanders.

Dinur and Meshulam [20] show that there exists a test that for any simplicial complex  $X$  and an alleged cover given by a simplicial map  $\rho : Y \rightarrow X$  samples  $q$  points  $(u_i, \rho(u_i))$  and measures how close  $\rho$  is to an actual covering map. The query complexity of the test is  $q = 3t$  points. Its soundness is affected by the cosystolic expansion of 1-chains. Using our new bounds on cosystolic expansion, we show that the complexes constructed in [48] or in [42] are cover-stable, i.e. that there exists some universal constant  $c > 0$ , such that for every  $\rho : Y(0) \rightarrow X(0)$

$$\mathbb{P}_{(u_i, \rho(u_i))_{i=1}^q} [\text{test fails}] \geq c \cdot \min \{ \text{dist}(\rho, \psi) \mid \psi : Y(0) \rightarrow X(0) \text{ is a cover} \},$$

where the distance is Hamming distance.

### Agreement testing

Coboundary expansion found an exciting new application in agreement testing [28, 11, 5]. An agreement test is a consistency test that originated as a component in low degree testing [58, 2, 56], but has been extensively studied ever since (see e.g. [26, 34, 18]). This test is a crucial component in many PCP constructions [57, 26, 15, 34, 19]. Given a set of partial functions on a set, an agreement test is a way to test whether these functions are correlated with some function that defined on the whole vertex set. The works [28, 11, 5] mentioned above use coboundary expansion to characterize when an agreement test is sound. Via this characterization they analyze agreement tests on high dimensional expanders. Continuing this line of works, [12, 14, 4] use theorems and tools developed in a preliminary version of this paper, to lower bound coboundary expansion of new high dimensional expanders, and with these lower bounds they obtain new agreement tests. These include the first agreement tests where the underlying complex family is bounded degree in the so called “list decoding regime” (the regime that is relevant to high-soundness PCPs such as the parallel repetition PCP [57, 34]).

---

<sup>3</sup> simplicial means that every  $i$ -face in  $Y$  is sent to an  $i$ -face in  $X$ .

## 1.2 Related work

Coboundary and Cosystolic expansion was defined by Linial, Meshulam and Wallach [45], [51], and independently by Gromov [30]. Gromov studied cosystolic expansion as a proxy for showing the topological overlapping property. Linial, Meshulam and Wallach were interested in analyzing high dimensional connectivity of random complexes.

Kaufman, Kazhdan and Lubotzky [35] introduced an elegant local to global argument for proving cosystolic expansion of 1-chains in the *bounded-degree* Ramanujan complexes of [49, 48]. This was significantly extended by Evra and Kaufman [22] to cosystolic expansion in all dimensions, thereby resolving Gromov’s conjecture about existence of bounded degree simplicial complexes with the topological overlapping property in all dimensions. Kaufman and Mass [37, 38] generalized the work of Evra and Kaufman from  $\mathbb{F}_2$  to all other groups as well, and used this to construct lattices with good distance. The best previously known bound for LSV complexes (1) was shown by Kaufman and Mass in [39].

Following ideas that appeared implicitly in Gromov’s work, Lubotzky Mozes and Meshulam analyzed the expansion of many “building like” complexes [47]. Kozlov and Meshulam [44] abstracted the main lower bound in [47] to the definition of cones (which they call chain homotopies), in order to analyze the coboundary expansion of geometric lattices and other complexes. Their work also connects coboundary expansion to other homological notions, and gives an upper bound to the coboundary expansion of bounded degree simplicial complexes. In [42], Kaufman and Oppenheim defined the notion of cones in order to analyze the cosystolic expansion of their high dimensional expanders (see [40]). In addition, they also come up with a criterion for showing that complexes admit short cones. They prove lower bounds on the cosystolic expansion of their complexes for 0- and 1-chains. The case of  $k$ -chains with  $k \geq 2$  is still open.

Several works tried to define quantum LDPC codes as cohomologies of simplicial complexes. Cosystolic expansion is used for analyzing the distance of the quantum code. Works by Evra, Kaufman and Zémor [23] and by Kaufman and Tessler [43] used cosystolic expansion in Ramanujan complexes to construct quantum codes that beat the  $\sqrt{n}$ -distance barrier. This continued in a sequence of works [54, 32, 7] which culminated in the breakthrough work of [55] that construct quantum LDPC codes with constant rate and distance. This later code is a cohomology of a certain chain complex, albeit not a simplicial complex; and it is analyzed essentially through the cosystolic expansion. Developing new techniques for cosystolic expansion can be potentially useful in this domain as well.

## 1.3 Open questions

The works by [47], [44] and [42] analyze a variety of symmetric complexes (that support a transitive group action). Could one combine our “color restriction” technique with the cone machinery to get lower bounds independent of degree and dimension on these complexes as well? There are a number of concrete constructions of local spectral high dimensional expanders that have excellent local spectral properties [8, 46, 27, 52, 9]. Are any of them cosystolic expanders?

Another intriguing direction of research is to develop additional techniques for analyzing coboundary or cosystolic expansion. The current techniques are limited to complexes that either have a lot of symmetry, or have excellent local expansion properties. Are there other complexes with these properties?

Our expansion bounds still have a dependence on the level ( $k$ ) of the chains. In the complete complex, for instance, this is not necessary. The complete complex is a  $\beta = 1 + o(1)$  coboundary expander for all  $k$ -chains [47]. It is not clear whether a dependence on  $k$  is necessary even in the spherical building. Which complexes have coboundary expansion that does not decay with the size of the chains?

Finally, the notion of coboundary and cosystolic expansion is closely related to locally testable codes and quantum LDPC codes. They also have connections to agreement expanders. It is interesting to find additional applications for these expanders.

## 1.4 Overview of the proof of Theorem 1

We start with a complex  $X$  that is a finite quotient of the affine building, as constructed by [48]. Our goal is to lower bound the cosystolic expansion of  $X$ . The proof has three components:

- (Theorem 2) A new local-to-global argument that derives cosystolic expansion of the complex from coboundary and spectral expansion of its links.
- (Theorem 3) A general color restriction technique that reduces the task of analyzing the coboundary expansion of a partite complex, to that of analyzing the local coboundary expansion of random color restrictions of it.
- (Theorem 4) Bounds on random color restrictions of (links of) the spherical building. Towards this end we construct a novel family of short cones for the spherical building (not based on apartments as in previous works [47]).

Below we give a short overview of each of these steps. For simplicity we assume in this subsection that  $\Gamma = \mathbb{F}_2$ , which captures the main ideas.

### The local to global argument, Theorem 2

Let  $X$  be our simplicial complex. We describe a correction algorithm, that takes as input a  $k$ -chain  $f : X(k) \rightarrow \mathbb{F}_2$ , with small coboundary  $\mathbb{P}[\delta f \neq 0] = \varepsilon$  and outputs a  $k$ -chain  $\tilde{f} : X(k) \rightarrow \mathbb{F}_2$  close to  $f$  that has no coboundary, i.e.  $\delta \tilde{f} = 0$ . For this overview, we focus on  $k = 1$ , i.e.  $f$  is a function on edges, which already exhibits the main ideas.

Let  $\eta > 0$  be some predetermined parameter. Our algorithm locally fixes “stars” of lower dimensional faces, that is, sets  $Star_k(r) = \{s \in X(k) \mid s \supseteq r\}$  for  $r \in X(j)$  (when  $j \leq k$ ). The fix takes place only if it is sufficiently useful: whenever it decreases the weight of  $\delta f$  by at least  $\eta \mathbb{P}[Star_k(r)]$ . In the case at hand,  $k = 1$ , so  $r$  is either a vertex or an edge, so

1. If  $r \in X(1)$ ,  $Star_1(r) = \{r\}$  and a fix just means changing the value of  $f(r)$ .
2. If  $r \in X(0)$ ,  $Star_1(r) = \{ru\}_{u \sim r}$  are all edges adjacent to  $r$ . Here a fix means changing the values of all  $\{f(ru) \mid u \sim r\}$  simultaneously.

#### ► Algorithm 5.

1. Set  $f_0 := f$ . Set  $i = 0$ .
2. While there exists a vertex or edge  $r \in X(0) \cup X(1)$  so that  $Star_k(r)$  has an assignment that satisfies a  $\eta \mathbb{P}[Star_k(r)]$ -fraction of faces more than the current assignment.
  - Let  $fix_r : Star_k(r) \rightarrow \Gamma$  be an optimal assignment to  $Star_k(r)$ .
  - Set  $f_{i+1}(s) = \begin{cases} f_i(t) & r \not\subseteq s \\ fix_r(s) & r \subseteq s \end{cases}$ .
  - Set  $i := i + 1$ .
3. Output the final function  $\tilde{f} := f_i$ .

The fact that we correct  $f$  locally only if the fix satisfies  $\eta$  fraction more triangles will promise that  $\text{dist}(f, \tilde{f}) \leq \frac{1}{\eta} \text{wt}(\delta f)$ . The output of the algorithm,  $\tilde{f}$ , is *not* necessarily locally minimal in the sense of [35, 22], but it is “ $\eta$ -locally-minimal”.

Notation: For functions  $g, h : X(\ell) \rightarrow \mathbb{R}$  we denote by  $\langle g, h \rangle = \mathbb{E}_{r \in X(\ell)} [g(r)h(r)]$  the usual inner product. For  $\ell = 1, 2$ , denote by  $D^\ell$  the *down operator* that takes  $h : X(2) \rightarrow \mathbb{R}$  and outputs  $D^\ell h : X(2 - \ell) \rightarrow \mathbb{R}$  via averaging. Namely  $D^\ell h(r)$  is the average of  $h(s)$  over  $s \supseteq r$ ,  $\mathbb{E}_{s \supseteq r} [h(s)]$ .

Let  $h : X(2) \rightarrow \mathbb{R}$  indicate the support of a  $\delta \tilde{f}$ , so  $h(t) = 1$  iff  $\delta \tilde{f} \neq 0$ . Our main argument is to show

$$\|D^3 h\|^2 \gtrsim \|D^2 h\|^2 \gtrsim \|Dh\|^2 \gtrsim \|h\|^2.$$

Eventually  $D^3 h = \mathbb{E}[h]^2$  is just a constant function. This shows that  $(\mathbb{E}[h])^2 = \text{const} \cdot \mathbb{E}[h]$  which implies that either the algorithm corrected  $f$  to a cosystol, i.e.  $h = 0$ , or that  $h$  has large weight, which implies that  $\delta f$  had large weight to begin with.

Let us show for example that  $\|D^3 h\|^2 \gtrsim \|D^2 h\|^2$  given that  $\|D^2 h\|^2 \gtrsim \|Dh\|^2 \gtrsim \|h\|^2$ . To do so, we define an auxiliary averaging operator  $N$  based on a random walk from vertices to triangles, and use the fact that in local spectral expanders,

$$\|D^3 h\|^2 \approx \langle Nh, D^2 h \rangle. \tag{2}$$

The operator  $N : \ell_2(X(2)) \rightarrow \ell_2(X(0))$  is defined by  $Nh(v) = \mathbb{E}_s [h(s)]$ , where  $s$  is sampled according to the following walk: Given  $v \in X(0)$ , sample some  $t \in X(3)$  such that  $v \in t$ , and then go to the triangle  $s = t \setminus \{v\}$ . We mention that the concept of localizing over such a distribution has appeared in [39]. The proof of (2) follows by localizing the expectation to the links and relying on the link expansion as in [53], [13, Claim 8.8] and in [41].

The key lemma in the proof shows that if there are many faces  $s' \supseteq v_0$  such that  $h(s') = 1$ , then there are many  $s$  such that  $v \notin s, \{v\} \cup s = t \in X(3)$ , where  $h(s) = 1$ . More precisely, we will show that for every  $v \in X(0)$  it holds that

$$Nh(v) \gtrsim \beta(D^2 h(v) - \eta)^4. \tag{3}$$

This immediately implies that

$$\begin{aligned} \langle Nh, D^2 h \rangle &= \mathbb{E}_v [D^2 h(v)Nh(v)] \\ &\stackrel{(3)}{\gtrsim} \beta(\mathbb{E}_v [(D^2 h(v))^2] - \eta \mathbb{E}_v [D^2 h(v)]) \\ &\gtrsim \beta\|D^2 h\|^2 - \beta\eta\|h\|^2 \\ &\gtrsim \beta\|D^2 h\|^2. \end{aligned}$$

The second inequality follows from  $\mathbb{E}_v [D^2 h(v)] = \mathbb{E}_s [h(s)] = \|h\|^2$ . The last inequality follows from the assumption that  $\|h\|^2 = O(\|D^2 h\|^2)$ . Combining this with (2) gives us the desired inequality.

Let us understand what is written in (3). On the right-hand side,  $D^2 h(v) = \mathbb{P}_{xy \in X_v(1)} [h(vxy) = 1]$  is the fraction of triangles  $vxy$  containing  $v$ , such that  $\delta \tilde{f}(vxy) \neq 0$ . On the left-hand side,  $Nh(v)$  is the fraction of  $s$  that complete  $v$  to some  $t = v \cup s \in X(3)$ , so that  $\delta \tilde{f}(s) \neq 0$ . For such an  $s = uxy$ ,

$$0 = \delta \delta \tilde{f}(vuxy) = \delta \tilde{f}(uxy) + (\delta \tilde{f}(vux) + \delta \tilde{f}(vuy) + \delta \tilde{f}(vxy)). \tag{4}$$

Set  $g : X_v(1) \rightarrow \mathbb{F}_2$  to be  $g(xy) = \delta \tilde{f}(vxy)$ , and note that  $g$  has the following properties:

1. By (4),  $\delta\tilde{f}(uxy) = 1 \iff \delta g(uxy) = 1$ .
2.  $\mathbb{P}[g \neq 0] = \mathbb{P}_{s \ni v}[\delta\tilde{f}(s) \neq 0] = D^2h(v)$ .
3.  $\eta$ -local-minimality: which is  $\text{dist}(g, B^1(X_v)) \geq \mathbb{P}[g \neq 0] - \eta$ , where  $B^1(X_v) = \{\delta\psi \mid \psi : X_v(0) \rightarrow \mathbb{F}_2\}$  is the set of coboundaries.

We explain the third item. Assume towards contradiction that  $\text{dist}(g, B^1(X_v)) < \mathbb{P}[g \neq 0] - \eta$  and let  $\delta\psi$  be a coboundary closest to  $g$ . Then by changing the values of  $\tilde{f}$  on  $\text{Star}(v)$  to be  $\tilde{f}'(vu) := \tilde{f}(vu) + \psi(u)$ , we have that whenever  $g(xy) = \delta\psi(xy)$ , then the fixed function satisfies  $\delta\tilde{f}'(vxy) = 0$ . I.e.

$$\text{dist}(g, \delta\psi) = \mathbb{P}_{vxy}[\delta\tilde{f}'(vxy) = 0] < \mathbb{P}_{vxy}[\delta\tilde{f}(vxy) = 0] - \eta.$$

This is a contradiction to the  $\eta$ -local minimality of  $\tilde{f}$  which is guaranteed by the algorithm.

Here is where the coboundary expansion of  $X_v$  comes into play. By coboundary expansion, we have that  $\mathbb{P}[\delta g(uxy) = 1] \geq \beta \text{dist}(g, B^1(X_v))$ . By combining the above we will get that

$$Nh(v) = \mathbb{P}_{uxy \in X_v(2)}[\delta\tilde{f}(uxy) \neq 0] \geq \beta \left( \mathbb{P}_{xy \in X_v(1)}[g(xy) \neq 0] - \eta \right) = \beta(D^2h(v) - \eta).$$

### The “color restriction” technique, Theorem 3

For this overview, assume that  $k = 2$ . The full details are in the full version of this paper [10]. Let  $Y$  be a  $d$ -dimensional  $(d+1)$ -partite complex so that a  $p$ -fraction of its color restrictions  $Y^F$  are  $\beta$ -local-coboundary expanders. We begin with a 2-chain  $f : Y(2) \rightarrow \mathbb{F}_2$  with small coboundary, namely  $\mathbb{P}_{s \in Y(3)}[\delta f(s) \neq 0] = \varepsilon$ . We need to find a 1-chain  $g : Y(1) \rightarrow \mathbb{F}_2$  so that  $\text{dist}(f, \delta g) \leq O(\frac{\varepsilon}{\beta^3 p})$ .

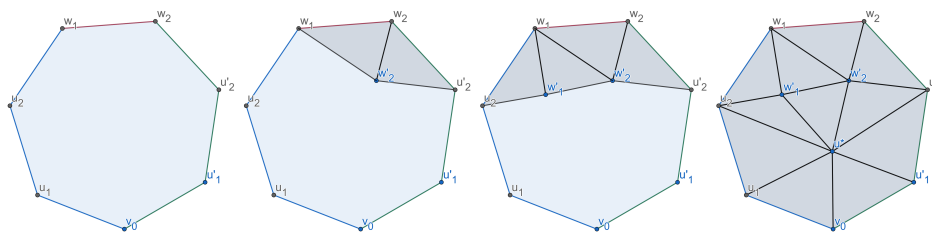
We first select a random color restriction, i.e. a set of colors so that  $Y^F$  is a local coboundary expander, that the weight of  $\delta f$  when restricted to triangles whose colors are in  $F$  is close to weight of  $\delta f$  on all  $Y$ . Averaging arguments guarantee that such  $F$  exists. Using this  $F$ , we construct  $g$  in three steps. In the first step we define  $g$  on edges with both endpoints colored in  $F$ ,  $uv \in Y^F$ . In the second step we define  $g$  on edges with one endpoint colored in  $F$ , i.e.  $uv \in Y(1)$  where  $u \in Y^F$  and  $v \notin Y^F$ . In the third step we define  $g$  on edges  $uv \in X(1)$  with neither endpoints colored in  $F$ , i.e. where  $u, v \notin Y^F$ . Every step uses the values of  $g$  that were constructed in the step before. For  $k > 2$  the  $(k-1)$ -chain is constructed following a similar sequence of  $k+1$  steps.

1. We start with the values of  $g$  on edges  $vu \in Y^F(1)$ . By the choice of  $F$ , the weight of  $\delta f$  inside  $Y^F$  is roughly  $\varepsilon$ . Local coboundary expansion implies that there exists a 1-chain  $g_0$  whose coboundary is close to  $f$  on  $Y^F$ . We set  $g(uv) = g_0(uv)$  for all  $uv \in Y^F(1)$ .
2. Next we define  $g$  on edges  $vu$  so that  $v \notin Y^F$  and  $u \in Y^F$ . Fix some  $v \notin Y^F$ . Let  $Y_v^F = \{s \in Y^F \mid s \cup v \in Y\}$ . This is the color restriction of the link of  $v$ . We wish to set values for  $g(vu)$  for all edges  $vu$  such that  $u \in Y_v^F(0)$ . We describe a system of equations that we use to set the values of  $g$  on the edges  $vu$  so as to satisfy a maximal number of equations. For every  $u_1u_2 \in Y_v^F(1)$ , the triangle  $vu_1u_2$  defines an equation:

$$f(vu_1u_2) + g(u_1u_2) = g(u_1v) + g(u_2v). \quad (5)$$

Note that the left-hand side of the equation is known since we have the values of  $f$  on all triangles, and we already constructed  $g$  for edges  $u_1u_2 \in Y^F(1)$ . So the above is an equation with two unknowns. We set  $g(vu)$  simultaneously for all  $u \in Y_v^F(1)$  to be an assignment that satisfies the largest fraction of equations (ties broken arbitrarily).

The idea behind this step is the following. Obviously, we'd like that  $f(vu_1u_2) = g(u_1u_2) + g(u_1v) + g(u_2v)$  for as many triangles as possible, so it makes sense to define  $g$  to satisfy



■ **Figure 1** Tiling a cycle.

the largest amount of equations (5). Let  $h_v : Y_v^F(1) \rightarrow \mathbb{F}_2$  be the left-hand side of (5), i.e.  $h_v(u_1u_2) = f(vu_1u_2) + g(u_1u_2)$ . We want to find an assignment  $g_v : Y_v^F(0) \rightarrow \mathbb{F}_2$  so that  $h_v(u_1u_2) = g_v(u_1) + g_v(u_2)$  for as many equations (5) as possible (and set  $g(vu) = g_v(u)$ ). Finding a solution  $g_v : Y_v^F(0) \rightarrow \mathbb{F}_2$  that satisfies (5) is equivalent finding  $g_v$  so that  $h_v(u_1u_2) = \delta g_v(u_1u_2)$ . Hence, to find an assignment that satisfies most of the equations is the same showing that  $h_v$  is close to a coboundary. In the analysis we show that  $\delta h_v \approx 0$ . This together with the local coboundary expansion of  $Y^F$  (which says that  $h^1(Y_v^F, \mathbb{F}_2) \geq \beta$ ) will show that indeed we can find satisfying  $\{g_v\}_{v \notin Y^F}$  so that  $f \approx \delta g$  where the distance is over edges  $uv$  where  $v \notin Y^F, u \in Y^F$ .

3. Finally we need to define the values of  $g$  on edges  $vu$  so that  $v, u \notin Y^F$ . Let  $vu$  be such an edge. Every triangle  $uvw$  where  $w \in Y_{vu}^F(0)$  defines a constraint on  $g(vu)$ :

$$f(uvw) + g(uw) + g(vw) = g(vu). \tag{6}$$

As in the previous case,  $f(uvw)$  is known, and  $g(uw), g(vw)$  were determined in step 2. We set  $g(vu) = \text{maj} \{f(uvw) + g(uw) + g(vw) \mid w \in Y_{vu}^F(0)\}$ . Ties are broken arbitrarily. Here we use the local coboundary expansion of  $Y^F$  in a way similar to the previous step, to show that indeed  $f \approx \delta g$ .

### New bounds on color-restrictions of the spherical building via cones, Theorem 4

In order to apply the color restriction technique we need to show that for a  $d$ -dimensional spherical building, many color restrictions are coboundary expanders<sup>5</sup>. For this overview we assume that  $k = 1$  and  $|F| = 5$ . Let us see how to bound coboundary expansion by constructing short cones.

It turns out easier to do so when the set of colors is a set of colors that are geometrically increasing (e.g. for  $k = 1$  we need colors  $F = \{i_1, i_2, \dots, i_5\}$  so that  $i_j \geq 10i_{j-1}$ ). The fraction of such sets of colors  $F$  is a constant that doesn't depend on  $d$  (it may depend on  $k$ ). For example, there is a constant probability that we select colors  $F$  so that for  $j = 1, 2, \dots, 5$ ,  $\frac{d}{10^{16-3j}} \leq i_j < \frac{2d}{10^{16-3j}}$ , since each of these intervals are a constant fraction of the interval  $[1, 2, \dots, d]$ . When these inequalities hold then  $i_j \geq 10i_{j-1}$ .

Denote by  $Y$  the  $SL_d(\mathbb{F}_q)$ -spherical building. Let  $Y^F$  be a complex induced by the subspaces of dimensions (i.e., colors)  $F = \{i_1, i_2, \dots, i_5\}$  so that  $i_j \geq 10i_{j-1}$ . Using the cone technology described in the full version of this paper [10], showing the  $Y^F$  is a coboundary expander reduces to showing that there is a short 1-cone on  $Y^F$ . A 1-cone consists of three things:

<sup>5</sup> In fact, we need to show that the links of the color restrictions are also coboundary expanders, but we ignore this point in the overview for brevity.

1. A vertex  $v \in X(0)$  (sometimes called the apex).
  2. For every  $u$ , a path  $p_u$  from the apex  $v$  to  $u$  in  $Y^F(1)$ .
  3. For every edge  $uw \in$ , a tiling by triangles  $t_{uw} \subset Y^F(2)$  of the cycle that consists of the path  $p_u$  from  $v$  to  $u$ , the edge  $uw$  and the path  $p_w$  from  $w$  back to  $v$ . Denote this cycle by  $p_u \circ uw \circ p_w$ . Here a tiling is a set of triangles whose boundary is the edges of the cycle.
- We give a formal and general definition of cones in the full version of this paper [10]. The radius of a cone is  $\text{rad}((v, \{p_u\}_{u \in Y^F(0)}, \{t_{uw}\}_{uw \in Y^F(1)})) = \max_{uw \in X(1)} |t_{uw}|$ .

We start by choosing an apex  $v = v_0$  of dimension  $i_1$  arbitrarily. Next we choose our paths to be as short as possible, and to consist of subspaces of dimension as low as possible. Explicitly we do the following.

1. For  $u$  adjacent to  $v_0$ , set  $p_u = (v_0, u)$ .
2. For  $u$  of the same dimension as  $v_0$  we find some  $w$  of dimension  $i_2$  so that  $w$  is a neighbour of  $v_0$  and  $u$ , and set  $p_u = (v_0, w, u)$ . This is always possible since the dimension of  $u + v_0$  is at most  $2i_1$ , so we can take any  $w$  of dimension  $i_2 \geq 2i_1$  that contains the sum of spaces. (Notice how the fact that dimensions are geometrically increasing is important here).
3. For other  $u \in Y^F(0)$ , we first take some  $w_2 \subseteq u$  of dimension  $i_1$ . Then we find some  $w_1$  who is a neighbour of  $v_0$  and of  $w_2$  and we set  $p_u = (v_0, w_1, w_2, u)$ .

Constructing  $t_{w_1 w_2}$  requires more care. Let us first consider the easier case. If  $\dim(w_1), \dim(w_2) \leq i_4$  then the cycle  $p_{w_1} \circ w_1 w_2 \circ p_{w_2}$  contains at most 7 vertices, all of dimension  $\leq i_4$ . In particular, the sum of all the vertices/subspaces is of dimension at most  $7i_4 \leq i_5$ , so there is a vertex  $u^*$  of dimension  $i_5$  that contains all the vertices in the cycle. The set of triangles  $u^*xy$  for all edges  $xy$  in the cycle is indeed a tiling of the cycle.

In the general case, it could be that the dimension of (say)  $w_1$  is  $i_5$ . For example, assume that  $\dim(w_1) = i_5, \dim(w_2) = i_4$  (in particular  $w_2 \subseteq w_1$ ). It is useful to read this description while looking at Figure 1. In this case, we first find a tiling that “shifts” the cycle to a cycle of low dimension vertices. More explicitly, we find some  $w'_2 \subseteq w_2$  of dimension  $i_3$ , that is also connected to  $w$ ’s neighbours in the cycle. These neighbours are  $w_1$  (and any subspace of  $w_2$  is connected to it), and some  $u'_2$  of dimension  $\leq i_2$ , so we can indeed find some  $w'_2$  that is connected to  $u$  and  $u'_2$  of dimension  $i_3$ . We tile the cycle with  $w_2 w'_2 u'_2, w_2 w'_2 w_1$ . This exchanges  $w_2$  with  $w'_2$  in the untiled cycle. We perform a similar vertex-switch, for  $w_1$  as well, finding some  $w'_1$  of dimension  $i_4$  that is connected to  $w_1$  neighbours in the untiled cycle. After these two steps, we can find a  $u^*$  that is connected to all the (now low-dimensional) cycle as in the previous case.

## 1.5 Organization of this paper and the full version

Section 2 contains preliminaries. We prove Theorem 2 that connects coboundary expansion in links to cosystolic expansion in Section 3 via the local correction algorithm. We develop the “color restriction” technique and prove Theorem 3 in the full version of this paper [10]. We analyze the expansion of the spherical building and other homogeneous geometric lattices in the full version of this paper [10]. In the full version [10], we also tie everything up and prove Theorem 1, as well as present the aforementioned applications of this bound. We also give there an upper bound on the cosystolic expansion of dense complexes.

## 2 Preliminaries and notation

For a more thorough preliminary section, see the full version of the paper [10].



### Simplicial complexes

A pure  $d$ -dimensional simplicial complex  $X$  is a set system (or hypergraph) consisting of an arbitrary collection of sets of size  $d + 1$  together with all their subsets. The sets of size  $i + 1$  in  $X$  are denoted by  $X(i)$ , and in particular, the vertices of  $X$  are denoted by  $X(0)$ . We will sometimes omit set brackets and write for example  $uvw \in X(2)$  instead of  $\{u, v, w\} \in X(2)$ . As convention  $X(-1) = \{\emptyset\}$ . Unless it is otherwise stated, we always assume that  $X$  is finite. Let  $X$  be a  $d$ -dimensional simplicial complex. Let  $k \leq d$ . We denote the set of oriented  $k$ -faces in  $X$  by  $\vec{X}(k) = \{(v_0, v_1, \dots, v_k) \mid \{v_0, v_1, \dots, v_k\} \in X(k)\}$ . For  $s = (v_0, v_1, \dots, v_k) \in \vec{X}(k)$  we denote  $set(s) = \{v_i\}_{i=0}^k$ , but when its clear from context we abuse notation and write  $s$  for its underlying set instead of  $set(s)$ . For an oriented face  $s \in \vec{X}(k)$  and an index  $i \in \{0, 1, \dots, k\}$ , we denote by  $s_i$  the face obtained by removing the  $i$ -th vertex of  $s$ .

Finally, Let  $s = (v_0, \dots, v_i)$ , and  $t = (u_0, \dots, u_j)$ . We denote by the concatenation  $s \circ t = (v_0, v_1, \dots, v_i, u_0, u_1, \dots, u_j)$ .

### Probability over simplicial complexes

Let  $X$  be a simplicial complex and let  $\mathbb{P}_d : X(d) \rightarrow (0, 1]$  be a density function on  $X(d)$  (i.e.  $\sum_{s \in X(d)} \mathbb{P}_d(s) = 1$ ). This density function induces densities on lower level faces  $\mathbb{P}_k : X(k) \rightarrow (0, 1]$  by  $\mathbb{P}_k(t) = \frac{1}{\binom{d+1}{k+1}} \sum_{s \in X(d), s \supset t} \mathbb{P}_d(s)$ . We can also define a probability over directed faces, where we choose an ordering uniformly at random. Namely, for  $s \in \vec{X}(k)$ ,  $\mathbb{P}_k(s) = \frac{1}{(k+1)!} \mathbb{P}_k(set(s))$ . When it's clear from the context, we omit the level of the faces, and just write  $\mathbb{P}[T]$  or  $\mathbb{P}_{t \in X(k)}[T]$  for a set  $T \subseteq X(k)$ .

## 2.1 Coboundary and cosystolic expansion

### Asymmetric functions

Let  $X$  be a  $d$ -dimensional simplicial complex. Let  $-1 \leq k \leq d$  be an integer. Let  $\Gamma$  be a group. A function  $f : \vec{X}(k) \rightarrow \Gamma$  is *asymmetric* if for every  $(v_0, v_1, \dots, v_k) \in \vec{X}(k)$ , and every permutation  $\pi : [k] \rightarrow [k]$  it holds that

$$f(v_0, v_1, \dots, v_k) = f(v_{\pi(0)}, v_{\pi(1)}, \dots, v_{\pi(k)})^{sign(\pi)}.$$

We denote the set of these functions by  $C^k(X, \Gamma)$ . We note that by fixing some order to the vertices  $X(0) = \{v_0, v_1, \dots, v_n\}$ , there is a bijection between functions  $f : X(k) \rightarrow \Gamma$  and asymmetric functions  $\vec{f} : \vec{X}(k) \rightarrow \Gamma$ . Given  $f : X(k) \rightarrow \Gamma$  and a set  $s = \{v_{i_0}, v_{i_1}, \dots, v_{i_k}\}$  so that  $i_0 < i_1 < \dots < i_k$ , we set  $\vec{f}(v_{\pi(i_0)}, v_{\pi(i_1)}, \dots, v_{\pi(i_k)}) = f(s)^{sign(\pi)}$ .

Let  $f : \vec{X}(k) \rightarrow \Gamma$ . The weight of  $f$  is  $wt(f) = \mathbb{P}_{t \in X(k)}[f(t) \neq 0]$ . For two functions  $f, g : \vec{X}(k) \rightarrow \Gamma$  the distance between  $f$  and  $g$  is  $dist(f, g) = wt(f - g) = \mathbb{P}_{t \in X(k)}[f(t) \neq g(t)]$ .

### Cohomology

Let  $\Gamma$  be an abelian group. The coboundary operator  $\delta_k : C^k(X, \Gamma) \rightarrow C^{k+1}(X, \Gamma)$  is defined by  $\delta_k f(s) = \sum_{i=0}^k (-1)^i f(s_i)$ . It is a direct calculation to verify that for any asymmetric function  $f \in C^k$  the function  $\delta_k f$  is indeed an asymmetric function, and that  $\delta_{k+1} \circ \delta_k = 0$ .

## 62:14 Coboundary and Cosystolic Expansion Without Dependence on Dimension or Degree

Let  $B^k(X, \Gamma) = \text{Im}(\delta_{k-1})$  be the space of coboundaries. Let  $Z^k(X, \Gamma) = \text{Ker}(\delta_k)$  be the space of cosystols. As  $\delta_{k+1} \circ \delta_k = 0$ , it holds that  $B^k(X, \Gamma) \subseteq Z^k(X, \Gamma)$ . The  $k$ -cohomology is  $H^k(X, \Gamma) = Z^k(X, \Gamma)/B^k(X, \Gamma)$ .

### Coboundary expansion

For a function  $f : \vec{X}(k) \rightarrow \Gamma$  let  $\text{dist}(f, B^k) = \min_{g \in C^{k-1}} \text{dist}(f, \delta g)$ , be the minimal distance between  $f$  and a coboundary. The  $k$ -th coboundary constant of a complex  $X$  (with respect to an abelian group  $\Gamma$ ) is  $h^k(X, \Gamma) = \min_{f \in C^k \setminus B^k} \frac{wt(\delta f)}{\text{dist}(f, B^k)}$  where  $B^k = B^k(X, \Gamma)$ . Note that  $h^k(X, \Gamma) > 0$  if and only if  $H^k = 0$ .

### Cosystolic expansion

A very related high dimensional notion of expansion is cosystolic expansion. The  $k$ -th cosystolic expansion constant of  $X$  (with respect to an abelian group  $\Gamma$ ) is

$$h^k(X, \Gamma) = \min_{f \in C^k \setminus Z^k} \frac{wt(\delta f)}{\text{dist}(f, Z^k)},$$

where  $Z^k = Z^k(X, \Gamma)$ . Notice that when  $B^k(X, \Gamma) = Z^k(X, \Gamma)$ , namely, when  $H^k = 0$ , this coincides with the definition of coboundary expansion, and this justifies using the same notation  $h^k$ , where the term coboundary expansion (as opposed to cosystolic expansion) is taken to indicate  $H^k = 0$ .

Another useful way to understand the constant is the following.  $h^k(X, \Gamma) \geq \beta$  if and only if for every  $f : \vec{X}(k) \rightarrow \Gamma$  there is some  $h \in Z^k(X, \Gamma)$  so that  $\beta \text{dist}(f, h) \leq wt(\delta f)$ . We note that in the work of [22] cosystolic expanders were also required to have no small weight  $f \in Z^k(X, \Gamma) \setminus B^k(X, \Gamma)$ . We don't focus on this notion in our work.

### Non abelian coboundary and cosystolic expansion

For  $k = 0, 1$  we can define the cohomology with respect to non abelian groups as well. Let  $\Gamma$  be a non abelian group. As before, for every  $k$  we can define  $C^k(X, \Gamma)$ . We define the coboundary operators as follows:

1.  $\delta_{-1} : C^{-1}(X, \Gamma) \rightarrow C^0(X, \Gamma)$  is  $\delta_{-1}h(v) = h(\emptyset)$ .
2.  $\delta_0 : C^0(X, \Gamma) \rightarrow C^1(X, \Gamma)$  is  $\delta_0h(vu) = h(v)h(u)^{-1}$ .
3.  $\delta_1 : C^1(X, \Gamma) \rightarrow C^2(X, \Gamma)$  is  $\delta_1h(vuw) = h(vu)h(uw)h(wv)$ .

It is easy to check that  $\delta_{k+1} \circ \delta_k f = e$  where  $e \in \Gamma$  is the unit. The definitions for  $h^k(X, \Gamma)$  and coboundary expansion are the same as in the abelian case for  $k = 0, 1$ .

## 2.2 Local properties of simplicial complexes

### Links of faces

Let  $X$  be a  $d$ -dimensional simplicial complex. Let  $k < d$  and  $s \in X(k)$ . The link of  $s$  is a  $d - k - 1$ -dimensional simplicial complex defined by  $X_s = \{t \setminus s \mid t \in X, t \supseteq s\}$ . We point out that the link of the empty set is  $X_\emptyset = X$ . Let  $s \in X(k)$  for some  $k \leq d$ . The density function  $\mathbb{P}_d$  on  $X$  induces on the link is  $\mathbb{P}_{d-k-1}^s : X(d - k - 1) \rightarrow (0, 1]$  where  $\mathbb{P}_{d-k-1}^s[t] = \frac{\mathbb{P}[t \cup s]}{\mathbb{P}[s] \binom{d+1}{k+1}}$ . We usually omit  $s$  in the probability, and for  $T \subseteq X_s(k)$  we write  $\mathbb{P}_{t \in X_s(k)}[T]$  instead.

### High dimensional local spectral expanders

Let  $X$  be a  $d$ -dimensional simplicial complex. Let  $k \leq d$ . The  $k$ -skeleton of  $X$  is  $X^{\leq k} = \bigcup_{j=-1}^k X(j)$ . In particular, the 1-skeleton of  $X$  is a graph.

► **Definition 6** (Spectral expander). Let  $G = (V, E)$  be a graph (that is, a 1-dimensional simplicial complex). Let  $A$  be its normalized adjacency operator, i.e. for every  $f : V \rightarrow \mathbb{R}$ ,  $Af : V \rightarrow \mathbb{R}$  is the function  $Af(v) = \mathbb{E}_{uv \in E} [f(u)]$ . Let  $1 = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{|V|} \geq -1$  be the eigenvalues of  $A$ .

Let  $\lambda \geq 0$ . We say that  $G$  is a  $\lambda$ -one sided spectral expander if  $\lambda_2 \leq \lambda$ . We say that  $G$  is a  $\lambda$ -two sided spectral expander if  $\lambda_2 \leq \lambda$  and  $\lambda_{|V|} \geq -\lambda$ .

► **Definition 7** (high dimensional local spectral expander). Let  $X$  be a  $d$ -dimensional simplicial complex. Let  $\lambda \geq 0$ . We say that  $X$  is a  $\lambda$ -one sided (two sided) local spectral expander if for every  $s \in X^{\leq d-2}$ , the 1-skeleton of  $X_s$  is a  $\lambda$ -one sided (two sided) spectral expander.

### 3 Cosystolic expansion

In this section we prove that local spectral expanders whose links are coboundary expanders are cosystolic expanders, that is, Theorem 2.

In fact, we prove a slightly more general statement, allowing for different coboundary expansion in every level.

► **Theorem 8.** Let  $k > 0$  be an integer and let  $\beta_0, \beta_1, \beta_2, \dots, \beta_k \in (0, 1]$  and  $\lambda > 0$ . Let  $X$  be a  $d$ -dimensional simplicial complex for  $d \geq k + 2$  and assume that  $X$  is a  $\lambda$ -one-sided local spectral expander. Let  $\Gamma$  be any group. Assume that for every  $0 \leq \ell \leq k$  and  $r \in X(\ell)$ ,  $X_r$  is a coboundary expander and that  $h^{k-\ell}(X_r, \Gamma) \geq \beta_{k-\ell}$ . Then  $h^k(X, \Gamma) \geq \frac{\prod_{\ell=0}^k \beta_\ell}{(k+2)! \cdot 4} - e\lambda$ .

Here  $e \approx 2.71$  is Euler's number. Obviously, Theorem 2 follows from Theorem 8 by setting  $\beta_\ell = \beta$  for every  $\ell = 0, 1, 2, \dots, k$ . The following proposition, that is important for the topological overlapping property will also be proven via similar arguments.

► **Proposition 9.** Let  $k > 0$  be an integer and let  $\beta_0, \beta_1, \beta_2, \dots, \beta_{k-1} \in (0, 1]$  and  $\lambda > 0$ . Let  $X$  be a  $d$ -dimensional simplicial complex for  $d \geq k + 1$  and assume that  $X$  is a  $\lambda$ -one-sided local spectral expander. Let  $\Gamma$  be any group. Assume that for every  $0 \leq \ell \leq k - 1$  and  $r \in X(\ell)$ ,  $X_r$  is a coboundary expander and that  $h^{k-\ell}(X_r, \Gamma) \geq \beta_{k-\ell-1}$ . Then every  $g \in Z^k(X, \Gamma) \setminus B^k(X, \Gamma)$ , has  $wt(g) \geq \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - e\lambda$ .

We remark that the when  $\Gamma$  is non abelian, these statements make sense only when  $k = 1$ . Turning back to Theorem 8, we present a correction algorithm. We will show that when  $f \in C^k(X, \Gamma)$  has a small coboundary, then the algorithm below returns some  $\tilde{f} \in Z^k(X, \Gamma)$  that is close to  $f$ .

► **Algorithm 10.** Input: A function  $f : \vec{X}(k) \rightarrow \Gamma$ , a parameter  $\eta \leq 1$ . Output: A function  $\tilde{f} : \vec{X}(k) \rightarrow \Gamma$ .

1. Set  $f_0 := f$ . Set  $i = 0$ .
2. While there exists  $\ell \leq k$ , and a face  $r \in \vec{X}(\ell)$  so that  $Star_k(r) = \{s \in X(k) \mid r \subseteq s\}$  has an assignment that satisfies a  $\eta \mathbb{P}[Star_k(r)]$ -fraction of faces more than the current assignment, do:
  - Let  $fix_r : Star_k(r) \rightarrow \Gamma$  be an optimal assignment to  $Star_k(r)$ , satisfying the maximal number of  $k + 1$ -faces containing  $r$ .
  - Set  $f_{i+1}(s) = \begin{cases} f_i(s) & r \not\subseteq s \\ fix_r(s) & r \subseteq s \end{cases}$ .
  - Set  $i := i + 1$ .
3. Output  $\tilde{f} := f_i$ .

### 3.1 Properties of Algorithm 10

Before proving Theorem 8 we record some properties of Algorithm 10.

▷ Claim 11. Algorithm 10 halts on every input.

▷ Claim 12. Let  $f : \vec{X}(k) \rightarrow \Gamma$  and let  $\eta \leq 1$ . Let  $\tilde{f} : \vec{X}(k) \rightarrow \Gamma$  be the output of Algorithm 10 on  $(f, \eta)$ . Then  $\eta \text{dist}(f, \tilde{f}) \leq \text{wt}(\delta f)$ .

These claims are elementary, they proven in full in the full version of this paper [10].

### 3.2 Local minimality

► **Definition 13** (Restriction). Let  $g \in C^k(X, \Gamma)$  and let  $r \in X(\ell)$  for some  $0 \leq \ell \leq k - 1$ . The restriction of  $g$  to  $r$  is the function  $g_r \in C^{k-\ell-1}(X_r, \Gamma)$  is defined by  $g_r(p) = g(r \circ p)$ .

► **Definition 14** (Local minimality). Let  $\eta \geq 0$  and let  $g \in C^k(X, \Gamma)$ . We say that  $g$  is  $\eta$ -locally minimal, if for every  $0 \leq \ell \leq k - 1$ , every  $r \in X(\ell)$ , and every  $h \in C^{k-\ell-2}(X_r, \Gamma)$  it holds that  $\text{wt}(g_r) \leq \text{wt}(g_r + \delta h) + \eta$ .

► **Definition 15** (Non abelian local minimality). If  $\Gamma$  is non-abelian we need the correct analogy to adding coboundaries. The definition of  $\eta$ -minimality is as follows. If  $k = 1$ , we say that  $g$  is  $\eta$ -locally minimal if for every  $v \in X(0)$ , and every  $\gamma \in \Gamma$ , it holds that  $\text{wt}(g_v) \leq \text{wt}(\gamma \cdot g_v) + \eta$ . If  $k = 2$ , we say that  $g$  is  $\eta$ -locally minimal if:

1. For every edge  $uv$  and every  $\gamma \in \Gamma$ , it holds that  $\text{wt}(g_{uv}) \leq \text{wt}(\gamma \cdot g_{uv}) + \eta$ .
2. For every vertex  $v$  and every function  $h : X_v(0) \rightarrow \Gamma$ , it holds that  $\text{wt}(g_v) \leq \text{wt}(g_v^h) + \eta$ , where  $g_v^h(uw) = h^{-1}(u)g_v(uw)h(w)$ .

The following claim is standard and is proven in the full version of this paper [10]

▷ Claim 16. Let  $f : \vec{X}(k) \rightarrow \Gamma$  and let  $\eta \leq 1$ . Let  $\tilde{f} : \vec{X}(k) \rightarrow \Gamma$  be the output of Algorithm 10 on  $(f, \eta)$ . Then  $\delta \tilde{f}$  is  $\eta$ -locally minimal.

### 3.3 Locally minimal cosystols are heavy

The following lemma states that non-zero functions that are locally minimal must have large weight.

► **Lemma 17.** Let  $\beta_0, \dots, \beta_{k-1}$  and  $\lambda$  be as in Theorem 8. Let  $X$  be such that for every  $0 \leq \ell \leq k - 1$  and every  $s \in X(\ell)$  it holds that  $X_s$  is a coboundary expander and  $h^{k-\ell-1}(X_s, \Gamma) \geq \beta_{k-\ell-1}$ . Assume further that  $X$  is a  $\lambda$ -local spectral expander. Let  $g \in Z^k(X, \Gamma)$  be non-zero and  $\eta$ -locally minimal. Then

$$\text{wt}(g) \geq \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - e(\eta + \lambda). \quad (7)$$

Additionally, for the case of non-abelian  $\Gamma$ , when  $k = 2$ , (7) holds for  $\eta$ -locally minimal and non-zero  $g = \delta f$ , for any  $f \in C^1(X, \Gamma)$ .

The last remark regarding  $k = 2$  is needed since  $Z^2(X, \Gamma)$  is not defined for non-abelian groups  $\Gamma$ . This lemma implies Theorem 2 and Proposition 9 directly.

**Proof of Theorem 8, given Lemma 17.** Fix  $\eta = \frac{\prod_{\ell=0}^k \beta_\ell}{4((k+2)!)}.$  Let  $\tilde{f}$  be the output of Algorithm 10 for some function  $f$  and  $\eta$ . If  $wt(\delta f) \geq \frac{\prod_{\ell=0}^k \beta_\ell}{4((k+2)!)} - e\lambda$  there is nothing to prove, so we assume that  $wt(\delta f) < \frac{\prod_{\ell=0}^k \beta_\ell}{4((k+2)!)} - e\lambda$ . Then  $\delta \tilde{f} \in Z^{k+1}(X, \Gamma)$  is an  $\eta$ -locally minimal function so that  $wt(\delta \tilde{f}) \leq wt(\delta f)$ . Hence by Lemma 17 (applied with  $k + 1$  instead of  $k$ ),  $\delta \tilde{f} = 0$  and  $\tilde{f}$  is a cosystol. By Claim 12,  $\eta \text{dist}(f, \tilde{f}) \leq wt(\delta f)$ , and we are done. ◀

**Proof of Proposition 9, given Lemma 17.** For every  $r \in X(j)$  and  $h \in C^{k-j-1}(X_r, \Gamma)$ , we define  $h^\uparrow : X(k) \rightarrow \Gamma$  by

$$h^\uparrow(s) = \begin{cases} h(p) & s = r \circ p \\ 0 & r \not\subseteq s. \end{cases}$$

It is easy to see that  $g_r + \delta h = (g + \delta h^\uparrow)_r$ .

Now let  $0 \neq g \in Z^k(X, \Gamma) \setminus B^k(X, \Gamma)$  be minimal among all  $Z^k(X, \Gamma) \setminus B^k(X, \Gamma)$ . By the above,  $g$  is also 0-locally minimal (since otherwise we could have found some non-zero coboundary  $\delta h^\uparrow$  to add to  $g$  and decrease its weight). Thus  $wt(g) \geq \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - e\lambda$  as required.

We remark that the case where  $\Gamma$  is non-abelian and  $k = 1$  is similar. Given  $g \in Z^1(X, \Gamma) \setminus B^1(X, \Gamma)$  that is non-zero and has minimal weight over all such functions. First we establish that it is locally minimal. Indeed, assume towards contradiction that there is some vertex  $v \in X(0)$  and  $\gamma \in \Gamma$  so that  $wt(g_v) < wt(\gamma g_v)$ . Then the function

$$g'(xy) = \begin{cases} \gamma g(xy) & x = v \\ g(xy)\gamma^{-1} & y = v \\ g(xy) & \text{otherwise} \end{cases}$$

is also a cosystol. Taking some triangle  $vuw \in X(2)$  that contains  $v$ , the value of

$$\delta g'(vuw) = \gamma \delta g(vuw) \gamma^{-1} = e$$

(the identity in  $\Gamma$ ). For any triangle  $uwx$  that doesn't contain  $v$  we have that  $\delta g'(uwx) = \delta g(uwx) = e$ . On the other hand,  $wt(g') < wt(g)$  so  $g'$  is trivial, which implies that  $g = \delta h$  where  $h(v) = \gamma$  and  $h(u) = e$ . A contradiction to the fact that  $g \notin B^1(X, \Gamma)$ . ◀

The remainder of this section is devoted to proving Lemma 17. For this we need to define averaging operators that play a crucial role in the theory behind local-spectral expanders. We will only define what we need so for a more thorough exposition see e.g. [13]. Let  $\ell_2(X(j))$  be the Hilbert space of all functions  $f : X(j) \rightarrow \mathbb{R}$  where the inner product is  $\langle f, g \rangle = \mathbb{E}_{r \in X(j)} [f(r)g(r)]$ . Let  $D_k : \ell_2(X(k)) \rightarrow \ell_2(X(k-1))$  be the following operator

$$D_k f(s) = \mathbb{E}_{t \supseteq s} [f(t)].$$

This operator's adjoint is  $U_{k-1} : \ell_2(X(k-1)) \rightarrow \ell_2(X(k))$  that is defined by

$$U_{k-1} f(t) = \mathbb{E}_{s \subseteq t} [f(s)].$$

As a shorthand we write  $D_k^\ell = D_{k-\ell+1} D_{k-\ell+2} \dots D_k$  for  $\ell \geq 1$  (and the same for  $U$ ). For  $\ell = 0$   $D_k^0 = U_k^0 = Id$ . We record that  $D_k^\ell f$  is a function whose domain is  $X(k-\ell)$ , and that  $U_k^\ell f$  is a function whose domain is  $X(k+\ell)$ .

Let  $j \leq k < d$ . The operator  $N_{k \rightarrow j} : \ell_2(X(k)) \rightarrow \ell_2(X(j))$  is defined by

$$N_{k \rightarrow j} f(r) = \mathbb{E}_{t \in X(k+1), t \supseteq r} \left[ \mathbb{E}_{s \subseteq t, r \not\subseteq s} [f(s)] \right].$$

Let us spell out this expression. We average over  $f(s)$  where  $s$  is chosen according to the following rule. We first sample some  $t \supseteq r$  in  $X(k+1)$ , and then we sample  $s \subseteq t$  given that it does not contain  $r$ .

When  $j, k$  is clear from the context we simply write  $D, U, N$ .

The following is an operator norm inequality that is similar to [13], but for the one-sided case. We prove it in the full version of this paper [10].

▷ **Claim 18.** Let  $X$  be a  $\lambda$ -one-sided local spectral expander. Then  $U_j^{k-j} N_{k \rightarrow j} \preceq U_{j-1}^{k-j+1} D_k^{k-j+1} + \lambda Id$  for every  $j \leq k$ .

Here  $A \preceq B$  for self adjoint operators  $A, B$  means that  $B - A$  is positive semi-definite, that is,  $\langle (B - A)h, h \rangle \geq 0$  for every function  $h$  in the domain of  $A, B$ .

**Proof of Lemma 17.** Let  $h = \mathbf{1}_{g \neq 0}$ . We will prove that  $wt(g) = \mathbb{E}[h] \geq \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - e(\eta + \lambda)$ . We do this by showing that

1.  $\|D_k h\|^2 \geq \frac{1}{k+1} \|h\|^2 - \lambda \|h\|^2$ .
2. For  $0 \leq j < k$ ,  $\|D_k^{k-j+1} h\|^2 \geq \frac{\beta_{k-j-1}}{j+1} \cdot \|D_k^{k-j} h\|^2 - \left( \frac{\beta_{k-j-1}\eta}{j+1} + \lambda \right) \|h\|^2$ .

We note that  $D^{k+1}h$  is a constant (as  $\lambda$ -local spectral expansion says in particular that the complex is connected) - the average of  $h$  on all faces. Hence  $\|D^{k+1}h\|^2 = \mathbb{E}[h]^2$ . By iteratively applying these inequalities we get that

$$\begin{aligned} \mathbb{E}[h]^2 &= \|D^{k+1}h\|^2 \\ &\geq \beta_{k-1} \|D^k h\|^2 - (\beta_{k-1}\eta + \lambda) \|h\|^2 \\ &\geq \frac{\beta_{k-1}\beta_{k-2}}{2} \|D^{k-1}h\|^2 - \beta_{k-1} \left( \frac{\beta_{k-2}\eta}{2} + \lambda \right) \|h\|^2 - (\beta_{k-1}\eta + \lambda) \|h\|^2 \\ &\dots \\ &\geq \|h\|^2 \cdot \left( \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - \eta \sum_{j=0}^{k-1} \frac{\beta_j}{(k-j+1)!} - \lambda \left( 1 + \sum_{j=0}^{k-1} \frac{\beta_j}{(k-j+1)!} \right) \right). \end{aligned}$$

By assuming  $\beta_j \leq 1$ , we upper bound  $\sum_{j=0}^{k-1} \frac{\beta_j}{(k-j+1)!} \leq \sum_{j=0}^{\infty} \frac{1}{j!} = e$ , and get  $\mathbb{E}[h]^2 \geq \|h\|^2 \cdot \frac{\prod_{\ell=0}^{k-1} \beta_\ell}{(k+1)!} - e(\eta + \lambda)$ . As  $\|h\|^2 = \mathbb{E}[h]$  the lemma follows.

Let us begin with the first item. we call  $s \in X(k)$  *active* if  $h(s) = 1$ . By assumption,  $g \in Z^k(X, \Gamma)$ , i.e.

$$\delta g(t) = \sum_{i=0}^{k+1} (-1)^i g(t_i) = 0.$$

Thus if  $t \in X(k+1)$  contains an active  $s = t_{i_1}$ , then it must also contain a second active

$s' = t_{i_2}$ <sup>6</sup>. This implies that  $N_{k \rightarrow k} h(s) \geq \frac{1}{k+1} h(s)$ , and so

$$\langle h, N_{k \rightarrow k} h \rangle = \mathbb{E}_t [h(t) N_{k \rightarrow k} h(t)] \geq \frac{1}{k+1} \|h\|^2.$$

By Claim 18  $N^{k \rightarrow k} \preceq UD + \lambda Id$ , so

$$\frac{1}{k+1} \|h\|^2 \leq \langle N_{k \rightarrow k} h, h \rangle \leq \langle UDh, h \rangle + \lambda \|h\|^2 = \|Dh\|^2 + \lambda \|h\|^2$$

so the first item is proven.

Next, we will prove the second item. As before, we will show that

$$\langle U^{k-j} N_{k \rightarrow j} h, h \rangle \geq \frac{\beta_{k-j-1}}{j+1} \cdot (\|D^{k-j} h\|^2 - \eta \|h\|^2). \quad (8)$$

Then we rely on Claim 18

$$\|D^{k-j+1} h\|^2 \geq \langle U^{k-j} N_{k \rightarrow j} h, h \rangle - \lambda \|h\|^2. \quad (9)$$

Combining these inequalities completes the proof.

We now state the following claim, which is proven using the coboundary expansion of  $X_r$  where  $r$  is a  $j$ -face.

► **Lemma 19** (Key lemma). *Let  $r \in X(j)$ . Then*

$$N_{k \rightarrow j} h(r) \geq \frac{\beta_{k-j-1}}{j+1} (D^{k-j} h(r) - \eta).$$

From this pointwise inequality, (8) follows easily:

$$\begin{aligned} \langle U^{k-j} N_{k \rightarrow j} h, h \rangle &= \langle N_{k \rightarrow j} h, D^{k-j} h \rangle \geq \mathbb{E}_r \left[ D^{k-j} h(r) \cdot \frac{\beta_{k-j-1}}{j+1} \cdot (D^{k-j} h(r) - \eta) \right] \\ &= \frac{\beta_{k-j-1}}{j+1} \cdot (\|D^{k-j} h\|^2 - \eta \|h\|^2) \end{aligned} \quad (10)$$

◀

We will prove Lemma 19 under the assumption that  $\Gamma$  is abelian since additive notation is more convenient. For non-abelian groups, see Remark 20.

**Proof of Lemma 19.** First, let us understand the meaning of the inequality in Lemma 19. Recall that  $N_{k \rightarrow j} h(r)$  is an average of  $h(s)$  over faces  $s \in X(k)$  so that  $r, s \subseteq t$  for some  $t \in X(k+1)$  and  $r \not\subseteq s$ . As  $h$  is an indicator function this is the same as writing

$$N_{k \rightarrow j} h(r) = \mathbb{P}_{t,s} [h(s) = 1],$$

where  $t, s$  are as above. On the other side of the inequality there is  $D^{k-j} h(r) = \mathbb{P}_{s \supseteq r} [h(s) = 1]$ . Hence, we need to show that if there are many active faces that contain  $r$ , there must also be many active faces that “complete”  $r$  to a  $(k+1)$ -face.

We first note that

$$N_{k \rightarrow j} h(r) = \mathbb{P}_{t,s} [h(s) = 1] \geq \frac{1}{j+1} \mathbb{P}_t [\exists s \subseteq t \ h(s) = 1 \text{ and } r \not\subseteq s], \quad (11)$$

so we shall actually lower bound  $\mathbb{P}_t [\exists s \subseteq t \ h(s) = 1 \text{ and } r \not\subseteq s]$ .

<sup>6</sup> in the case where  $\Gamma$  is non-abelian and  $g = \delta f \in C^2(X, \Gamma)$ , even though  $\delta g(abcd)$  is not defined, one still observes that  $\delta f(abc) = \delta f(acd) = \delta f(abd) = e$  implies that  $\delta f(bcd) = 0$  so the same conclusion holds.

As  $g \in Z^k(X, \Gamma)$ , for every  $t = r \circ p \in X(k+1)$

$$0 = \delta g(r \circ p) = \sum_{i=0}^j (-1)^i g(r_i \circ p) + (-1)^j \sum_{i=0}^{k-j} (-1)^i g(r \circ p_i). \quad (12)$$

And in particular

$$\sum_{i=0}^{k-j} (-1)^i g(r \circ p_i) \neq 0 \iff \sum_{i=0}^j (-1)^i g(r_i \circ p) \neq 0. \quad (13)$$

Recall that the restriction of  $g$  is  $g_r : X_r(k-j-1) \rightarrow \Gamma$ , defined by  $g_r(p) = g(r \circ p)$ . As we can see,  $\delta g_r(p)$  is the left-hand side of (13). Thus

$$\mathbb{P}_t [\exists s \subseteq t \ h(s) = 1 \text{ and } r \not\subseteq s] \geq \mathbb{P}_{t=r \circ p} \left[ \sum_{i=0}^{k-j} (-1)^i g(r \circ p_i) \neq 0 \right] \quad (14)$$

$$= \mathbb{P}_{p \in X_r(k-j)} [\delta g_r(p) \neq 0]. \quad (15)$$

By assumption  $X_r$  is a  $\beta_{k-j-1}$ -coboundary expander, this is at least  $\beta_{k-j-1} \cdot \text{dist}(g_r, B^{k-j-1}(X_r, \Gamma))$ .

To conclude we need to show that

$$\text{dist}(g_r, B^{k-j-1}(X_r, \Gamma)) \geq \mathbb{P}_{s \supseteq r} [g(s) \neq 0] - \eta. \quad (16)$$

But

$$\text{dist}(g_r, B^{k-j-1}(X_r, \Gamma)) = \min_{f \in C^{k-j-2}(X_r, \Gamma)} \{wt(g_r + \delta f)\} \geq wt(g_r) - \eta. \quad (17)$$

where the inequality follows from  $\eta$ -minimality of  $g$ . As  $wt(g_r) = \mathbb{P}_{s \supseteq r} [h(s) = 1]$  we have proven

$$N_{k \rightarrow j} h(r) \geq \frac{\beta_{k-j-1}}{j+1} \text{dist}(g_r, B^{k-j-1}(X_r, \Gamma)) \geq \frac{\beta_{k-j-1}}{j+1} \left( \mathbb{P}_{s \supseteq r} [h(s) = 1] - \eta \right). \quad \blacktriangleleft$$

► **Remark 20 (The non-abelian case).** The first place where we need to accommodate for the non-commutativity is in the derivation of (14). Let us understand how to substitute (12) which implies (13), for non-abelian groups.

If for example, if  $r \in X(0)$  and  $g \in Z^1(X, \Gamma)$ , and  $ruw \in X(2)$  we can write

$$e = \delta g(ruw) = g(ru)g(uw)g(wr)$$

instead of (12). This implies that

$$g(uw) = g(ur)g(rw) \quad (18)$$

or

$$g(rw)g(uw)g(wr) = g(rw) \cdot (g(ur)g(rw)) \cdot g(wr) = g(rw)g(ru)^{-1} = g_r(w)g_r(u)^{-1} = \delta g_r(wu)$$

where in the first equality we plugged in the first part of (18) and in the second to last equality we plugged in the second part of (18). Since the left hand side is a conjugation of  $g(uw)$ , we deduce that  $g(uw) \neq e \iff \delta g_r(wu) \neq e$ . This is the same conclusion as we get in (12). The case where  $r \in X(1)$  is similar.



If  $k = 2$  we cannot even define (12) since the coboundary map is not defined. Still, let us see that a similar conclusion to (13) holds. Let  $g = \delta f \in C^2(X, \Gamma)$ . Let  $r = ab \in X(1)$  and  $t = abcd \in X(3)$ . Denote by  $\gamma = f(ab)f(bc)f(cd)$ . Then

$$\begin{aligned} \gamma^{-1} \delta g_r(cd) \gamma &= \gamma^{-1} g(rc) g(rd)^{-1} \gamma \\ &= \gamma^{-1} g(abc) g(adb) \gamma \\ &= f(dc) \cdot \overline{(f(cb)f(ba)f(ab)f(bc))} \cdot f(ca) f(ad) f(db) \cdot \overline{(f(ba)f(ab))} \cdot f(bc) f(cd) \\ &= (f(dc) \overline{f(ca) f(ad)}) (f(db) f(bc) f(cd)) \\ &= \delta f(dca) \delta f(dbc) \\ &= g(dca) g(dcb)^{-1}. \end{aligned}$$

In particular, we deduce that  $g(dca)g(dcb)^{-1} \neq e \iff \delta g_r(cd) \neq e$ , and (14) now becomes

$$\mathbb{P}_t [\exists s \subseteq t \ h(s) = 1 \text{ and } r \not\subseteq s] \geq \mathbb{P}_{t=r \circ cd} [g(dca)g(dcb)^{-1} \neq 0] = \mathbb{P}_{cd \in X_r(1)} [\delta g_r(cd) \neq 0]. \quad (19)$$

Similarly, when  $r = a \in X(0)$  and  $t = abcd$  we observe similarly that

$$\begin{aligned} f(ab)g(bcd)f(ba) &= f(ab)(f(bc)f(cd)f(db))f(ba) \\ &= f(ab)f(bc) \cdot (f(ca)f(ac)) \cdot f(cd) \cdot (f(da)f(ad)) \cdot f(db)f(ba) \\ &= \delta f(abc) \delta f(acd) \delta f(adb) \\ &= g_a(bc) g_a(cd) g_a(db) \\ &= \delta g_a(bcd), \end{aligned}$$

and in particular

$$\mathbb{P}_t [\exists s \subseteq t \ h(s) = 1 \text{ and } r \not\subseteq s] \geq \mathbb{P}_{t=a \circ bcd} [g(bcd) \neq e] = \mathbb{P}_{bcd \in X_a(2)} [\delta g_a(bcd) \neq e]. \quad (20)$$

The second equality we need to modify is (17). For example, take an  $\eta$ -locally minimal  $g \in C^2(X, \Gamma)$ , a vertex  $r \in X(0)$ , and  $\delta h \in B^1(X_r, \Gamma)$  that is a closest coboundary to  $g_r \in C^1(X_r, \Gamma)$ . Then

$$\text{dist}(g_r, \delta h) = \mathbb{P} [g_r(vu) \neq h(v)h(u)^{-1}] = wt(g_r^h) \geq wt(g_r) - \eta.$$

The case where  $r \in X(1)$  is similar.

## References

- 1 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 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 1–12. ACM, 2019. doi:10.1145/3313276.3316385.
- 2 Sanjeev Arora and Madhu Sudan. Improved low degree testing and its applications. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 485–495, El Paso, Texas, 1997.
- 3 Mitali Bafna, Max Hopkins, Tali Kaufman, and Shachar Lovett. High dimensional expanders: Eigenstripping, pseudorandomness, and unique games. 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 1069–1128. SIAM, 2022. doi:10.1137/1.9781611977073.47.
- 4 Mitali Bafna, Noam Lifshitz, and Dor Minzer. Constant degree direct product testers with small soundness, 2024. arXiv:2402.00850.

- 5 Mitali Bafna and Dor Minzer. Characterizing direct product testing via coboundary expansion. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2024*, 2024.
- 6 Yehonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica*, 26(1439-6912):495–519, 2006. doi:10.1007/s00493-006-0029-7.
- 7 Nikolas P. Breuckmann and Jens N. Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, 2021. doi:10.1109/TIT.2021.3097347.
- 8 Michael Chapman, Nathan Linial, and Peled Yuval. Expander graphs — both local and global. *Combinatorica*, 40:473–509, 2020. doi:10.1007/s00493-019-4127-8.
- 9 Yotam Dikstein. New high dimensional expanders from covers. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*, pages 826–838, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585183.
- 10 Yotam Dikstein and Irit Dinur. Coboundary and cosystolic expansion without dependence on dimension or degree, 2023. arXiv:2304.01608.
- 11 Yotam Dikstein and Irit Dinur. Agreement theorems for high dimensional expanders in the small soundness regime: the role of covers. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2024*, 2024.
- 12 Yotam Dikstein and Irit Dinur. Swap cosystolic expansion. In *Proceedings of the 56th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2024*, 2024.
- 13 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *Proc. 20th International Workshop on Randomization and Computation (RANDOM)*, volume 116, pages 37:1–37:21, Princeton, NJ, 2018. RANDOM/APPROX. doi:10.4230/LIPIcs.APPROX/RANDOM.2018.37.
- 14 Yotam Dikstein, Irit Dinur, and Alexander Lubotzky. Low acceptance agreement tests via bounded-degree symplectic hdxs, 2024. arXiv:2402.01078.
- 15 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.
- 16 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. 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 357–374. ACM, 2022. doi:10.1145/3519935.3520024.
- 17 Irit Dinur, Yuval Filmus, Prahladh Harsha, and Madhur Tulsiani. Explicit sos lower bounds from high-dimensional expanders. In *12th Innovations in Theoretical Computer Science (ITCS 2021)*, 2020. URL: <https://arxiv.org/abs/2009.05218>.
- 18 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *Proc. 58th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 974–985, 2017. doi:10.1109/FOCS.2017.94.
- 19 Irit Dinur and Or Meir. Derandomized parallel repetition via structured pcps. *Comput. Complex.*, 20(2):207–327, 2011. doi:10.1007/s00037-011-0013-5.
- 20 Irit Dinur and Roy Meshulam. Near coverings and cosystolic expansion. *Archiv der Mathematik*, 118(5):549–561, May 2022. doi:10.1007/s00013-022-01720-6.
- 21 Dominic Dotterer, Tali Kaufman, and Uli Wagner. On expansion and topological overlap. *Geometriae Dedicata*, 195:307–317, 2018.
- 22 Shai Evra and Tali Kaufman. Bounded degree cosystolic expanders of every dimension. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, pages 36–48, 2016. doi:10.1145/2897518.2897543.
- 23 Shai Evra, Tali Kaufman, and Gilles Zémor. Decodable quantum LDPC codes beyond the square root distance barrier using high dimensional expanders. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 218–227. IEEE, 2020. doi:10.1109/FOCS46700.2020.00029.
- 24 Z. Füredi and J. Komlos. The eigenvalues of random symmetric matrices. *Combinatorica*, 1:233–241, 1981. doi:10.1007/BF02579329.

- 25 Jason Gaitonde, Max Hopkins, Tali Kaufman, Shachar Lovett, and Ruizhe Zhang. Eigenstripping, spectral decay, and edge-expansion on posets. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 16:1–16:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.16.
- 26 Oded Goldreich and Shmuel Safra. A combinatorial consistency lemma with application to proving the PCP theorem. In *RANDOM: International Workshop on Randomization and Approximation Techniques in Computer Science*. LNCS, 1997.
- 27 Louis Golowich. Improved Product-Based High-Dimensional Expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.38.
- 28 Roy Gotlib and Tali Kaufman. List agreement expansion from coboundary expansion, 2022. arXiv:2210.15714.
- 29 M. Gromov. Random walk in random groups. *Geom. Funct. Anal.*, 13:73–146, 2003. doi:10.1007/s000390300002.
- 30 M. Gromov. Singularities, expanders and topology of maps. part 2: from combinatorics to topology via algebraic isoperimetry. *Geom. Funct. Anal.*, 20:416–526, 2010. doi:10.1007/s00039-010-0073-8.
- 31 Tom Gur, Noam Lifshitz, and Siqi Liu. Hypercontractivity on high dimensional expanders. 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 176–184. ACM, 2022. doi:10.1145/3519935.3520004.
- 32 Matthew B. Hastings, Jeongwan Haah, and Ryan O’Donnell. Fiber bundle codes: breaking the  $n^{1/2}$  polylog( $n$ ) barrier for quantum LDPC codes. 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 1276–1288. ACM, 2021. doi:10.1145/3406325.3451005.
- 33 Max Hopkins and Ting-Chun Lin. Explicit lower bounds against  $\omega(n)$ -rounds of sum-of-squares. *arXiv preprint arXiv:2204.11469*, 2022.
- 34 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New direct-product testers and 2-query pcps. *SIAM Journal on Computing*, 41(6):1722–1768, 2012.
- 35 Tali Kaufman, David Kazhdan, and Alexander Lubotzky. Ramanujan complexes and bounded degree topological expanders. In *Proc. 55th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 484–493, 2014. doi:10.1109/FOCS.2014.58.
- 36 Tali Kaufman and Alexander Lubotzky. High dimensional expanders and property testing. In *Innovations in Theoretical Computer Science, ITCIS’14, Princeton, NJ, USA, January 12-14, 2014*, pages 501–506, 2014.
- 37 Tali Kaufman and David Mass. Good distance lattices from high dimensional expanders. (manuscript), 2018. arXiv:1803.02849.
- 38 Tali Kaufman and David Mass. Unique-Neighbor-Like Expansion and Group-Independent Cosystolic Expansion. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2021.56.
- 39 Tali Kaufman and David Mass. Double Balanced Sets in High Dimensional Expanders. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:17, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.3.
- 40 Tali Kaufman and Izhak Oppenheim. Construction of new local spectral high dimensional expanders. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 773–786, 2018.

- 41 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Comb.*, 40(2):245–281, 2020. doi:10.1007/s00493-019-3847-0.
- 42 Tali Kaufman and Izhar Oppenheim. Coboundary and cosystolic expansion from strong symmetry. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 84:1–84:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.84.
- 43 Tali Kaufman and Ran J. Tessler. New cosystolic expanders from tensors imply explicit quantum LDPC codes with  $\Omega(\sqrt{n} \log^k n)$  distance. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1317–1329. ACM, 2021. doi:10.1145/3406325.3451029.
- 44 Dmitry N Kozlov and Roy Meshulam. Quantitative aspects of acyclicity. *Research in the Mathematical Sciences*, 6(4):1–32, 2019.
- 45 Nathan Linial and Roy Meshulam. Homological connectivity of random 2-complexes. *Combinatorica*, 26:475–487, 2006. doi:10.1007/s00493-006-0027-9.
- 46 Siqi Liu, Sidhanth Mohanty, and Elizabeth Yang. High-Dimensional Expanders from Expanders. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics (LIPICs)*, pages 12:1–12:32, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPICs.ITCS.2020.12.
- 47 Alexander Lubotzky, Roy Meshulam, and Shahar Mozes. Expansion of building-like complexes. *Groups, Geometry, and Dynamics*, 10(1):155–175, 2016.
- 48 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type  $\tilde{A}_d$ . *European J. Combin.*, 26(6):965–993, 2005. doi:10.1016/j.ejc.2004.06.007.
- 49 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of type  $\tilde{A}_d$ . *Israel J. Math.*, 149(1):267–299, 2005. doi:10.1007/BF02772543.
- 50 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families i: Bipartite ramanujan graphs of all degrees. *Annals of Mathematics*, 182:307–325, 2015. doi:10.4007/annals.2015.182.1.7.
- 51 Roy Meshulam and N. Wallach. Homological connectivity of random  $k$ -dimensional complexes. *Random Struct. Algorithms*, 34(3):408–417, 2009. doi:10.1002/rsa.20238.
- 52 Ryan O’Donnell and Kevin Pratt. High-dimensional expanders from chevalley groups. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPICs*, pages 18:1–18:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.CCC.2022.18.
- 53 Izhar Oppenheim. Local spectral expansion approach to high dimensional expanders part I: Descent of spectral gaps. *Discrete Comput. Geom.*, 59(2):293–330, 2018. doi:10.1007/s00454-017-9948-x.
- 54 Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, pages 1–1, 2021. doi:10.1109/TIT.2021.3119384.
- 55 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. 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 375–388. ACM, 2022. doi:10.1145/3519935.3520017.
- 56 R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. 28th ACM Symp. on Theory of Computing*, pages 475–484, 1997.
- 57 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.
- 58 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

# Rapid Mixing of the Down-Up Walk on Matchings of a Fixed Size

Vishesh Jain ✉ 

Department of Mathematics, Statistics, and Computer Science,  
University of Illinois Chicago, Chicago, IL, USA

Clayton Mizgerd ✉

Department of Mathematics, Statistics, and Computer Science,  
University of Illinois Chicago, Chicago, IL, USA

---

## Abstract

Let  $G = (V, E)$  be a graph on  $n$  vertices and let  $m^*(G)$  denote the size of a maximum matching in  $G$ . We show that for any  $\delta > 0$  and for any  $1 \leq k \leq (1 - \delta)m^*(G)$ , the down-up walk on matchings of size  $k$  in  $G$  mixes in time polynomial in  $n$ . Previously, polynomial mixing was not known even for graphs with maximum degree  $\Delta$ , and our result makes progress on a conjecture of Jain, Perkins, Sah, and Sawhney [STOC, 2022] that the down-up walk mixes in optimal time  $O_{\Delta, \delta}(n \log n)$ .

In contrast with recent works analyzing mixing of down-up walks in various settings using the spectral independence framework, we bound the spectral gap by constructing and analyzing a suitable multi-commodity flow. In fact, we present constructions demonstrating the limitations of the spectral independence approach in our setting.

**2012 ACM Subject Classification** Mathematics of computing → Markov-chain Monte Carlo methods

**Keywords and phrases** Down-up walk, Matchings, MCMC

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.63

**Category** RANDOM

**Funding** *Vishesh Jain*: NSF CAREER award DMS-2237646

*Clayton Mizgerd*: NSF award ECCS-2217023

**Acknowledgements** We thank Huy Tuan Pham for helpful discussions and anonymous referees for several useful suggestions.

## 1 Introduction

Sampling and counting matchings in graphs is a central and well-studied problem. An early success in this direction is the classical algorithm of Kasteleyn for counting the number of perfect matchings in a planar graph [18]. Starting with the foundational work of Valiant [23], it was established that Kasteleyn’s algorithm is exceptional in the sense that it is #P-hard to (exactly) count the number of perfect matchings, even for restricted classes of input graphs such as bipartite graphs and graphs of bounded degree. In fact, perhaps quite surprisingly, the more general problem of counting matchings of a given size is #P-hard, even restricted to the class of planar graphs [15].

Given the above hardness results, the best one can hope for is fully polynomial-time (possibly randomized) approximation schemes. In particular, in connection with fully polynomial-time randomized approximation schemes (FPRAS) for the number of matchings (possibly of a given size), as well as being an important problem in its own right, much work has been devoted to the problem of approximately sampling from various distributions on matchings of a graph. The celebrated work of Jerrum and Sinclair [16] showed that for the monomer-dimer model at activity  $\lambda$  (i.e. the distribution on matchings where the probability



© Vishesh Jain and Clayton Mizgerd;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 63; pp. 63:1–63:13



Leibniz International Proceedings in Informatics

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

of a matching  $M$  is proportional to  $\lambda^{|M|}$ ;  $\lambda$  is known as the activity), the Glauber dynamics mixes in time polynomial in  $n$  and  $\lambda$ . For graphs  $G = (V, E)$  of bounded degree and  $\lambda = O(1)$ , the optimal mixing time  $O(|E| \log n)$  was obtained by Chen, Liu, and Vigoda [6].

By combining with a rejection sampling procedure, both of these works give polynomial time algorithms to approximately sample from the uniform distribution on matchings of size  $k \leq (1 - \delta)m^*(G)$  for any fixed  $\delta > 0$ , where  $m^*(G)$  denotes the matching number of  $G$  i.e. the size of a largest matching in  $G$ ; approximately sampling from the uniform distribution on perfect matchings of a graph remains a major open problem, although in the bipartite case, this was famously resolved by Jerrum, Sinclair, and Vigoda [17]. For the class of bounded degree graphs, an algorithm with near-optimal run time was provided by a recent work of Jain, Perkins, Sah, and Sawhney [14]; they gave an algorithm which, given a graph  $G$  of maximum degree  $\Delta$ , an integer  $1 \leq k \leq (1 - \delta)m^*(G)$ , and a parameter  $\varepsilon > 0$ , outputs a random matching  $M$  of size  $k$  in time  $\tilde{O}_{\Delta, \delta}(n)$ <sup>1</sup> such that the total variation distance is less than  $\varepsilon$  between the distribution on  $M$  and the uniform distribution on  $\mathcal{M}_k(G)$ : the matchings in  $G$  of size  $k$ .

Despite this progress, the mixing time of perhaps the simplest random walk on  $\mathcal{M}_k(G)$  – the so-called down-up walk – is not understood. By the down-up walk for matchings of size  $k$ , we refer to the following chain:

1. Denote the state at time  $t$  by  $M_t \in \mathcal{M}_k$ .
2. Choose  $e \in M_t$  and  $e' \in E$  uniformly at random.<sup>2</sup>
3. Let  $M' := M_t \cup \{e'\} \setminus \{e\}$ . If  $M' \in \mathcal{M}_k$ , then  $M_{t+1} = M'$ . Else,  $M_{t+1} = M_t$ .

It is clear that the down-up walk is reversible with respect to the uniform distribution on  $\mathcal{M}_k$  so that whenever it is ergodic (this need not be the case; for instance, consider the uniform distribution on perfect matchings of an even cycle), it converges to the uniform distribution on  $\mathcal{M}_k$ . It is believed that for any fixed  $\delta > 0$  the down-up walk on  $\mathcal{M}_k$  mixes in polynomial time for all  $1 \leq k \leq (1 - \delta)m^*(G)$ .<sup>3</sup> In fact, it was conjectured by Jain, Perkins, Sah, and Sawhney [14, Conjecture 1.4] that for graphs  $G$  of maximum degree  $\Delta$  and  $1 \leq k \leq (1 - \delta)m^*(G)$ , the  $\varepsilon$ -total-variation mixing time of the down-up walk on  $\mathcal{M}_k(G)$  is  $O_{\Delta, \delta}(n \log(n/\varepsilon))$ , which would be optimal up to the implicit constants.

The main result of this note establishes that the down-up walk on  $\mathcal{M}_k(G)$  mixes in polynomial time for all  $1 \leq k \leq (1 - \delta)m^*(G)$ . While our mixing time is unfortunately not sharp enough to resolve the aforementioned conjecture from [14], our result has the benefit of being applicable to arbitrary graphs (as opposed to graphs of bounded degree).

► **Theorem 1.** *Let  $\delta \in (0, 1)$ . For a graph  $G = (V, E)$  on  $n$  vertices and  $m$  edges, and an integer  $1 \leq k \leq (1 - \delta)m^*(G)$ , the down-up walk on matchings of size  $k$  has  $\varepsilon$ -mixing time  $O(n^{4/\delta} m^4 k \log(1/\varepsilon))$ .*

► **Remark 2.** Restricted to the class of graphs of maximum degree  $\Delta$ , our proof gives the improved  $\varepsilon$ -mixing time bound of  $O_{\Delta, \delta}(n^\delta k \log(1/\varepsilon))$  by Equation (7) and Equation (1). We leave it as a very interesting open problem whether the mixing time can be improved to  $\tilde{O}_{\Delta, \delta}(n)$  in this case (as was conjectured in [14]).

<sup>1</sup>  $\tilde{O}$  hides polylogarithmic factors in  $n$  and  $1/\varepsilon$ .

<sup>2</sup> Another convention is to choose  $e'$  uniformly at random among those edges for which  $M_t \cup \{e'\} \setminus \{e\} \in \mathcal{M}_k$ ; in our setting, this would only have the effect of leading to a constant factor speed-up in the mixing time.

<sup>3</sup> Some restriction on the range of  $k$  is needed since, as just mentioned, the down-up walk is not even ergodic in general.

Our proof is based on bounding the spectral gap using a carefully constructed flow. It is natural to ask whether the powerful spectral independence framework (developed in [3]) can be used to derive a similar result; in Section 3, we present examples showing that there are serious barriers to this, even for the class of bounded degree graphs. Roughly, the main point is that the condition  $k \leq (1 - \delta)m^*(G)$  is not closed under pinnings (even if we take pinnings at random); this is not the case for the parameter range of independent sets considered in [13] and is key to making the spectral independence approach amenable in their setting.

## 1.1 Related work

For the down-up walk, notice that even the case when  $G$  is itself a matching is already interesting; in this case, the down-up walk coincides with the classical and well-studied Bernoulli-Laplace chain to sample from the uniform distribution on  $\binom{[n]}{k}$  (e.g. [10, 19]). As discussed earlier, there are polynomial time algorithms, based on the rapid mixing of Glauber dynamics for the monomer-dimer model, to approximately sample from the uniform distribution on  $\mathcal{M}_k(G)$ ,  $1 \leq k \leq (1 - \delta)m_k^*(G)$ ; instead of combining rejection sampling with the Glauber dynamics, one may also combine rejection sampling with a local random walk to sample from the uniform distribution on the union of matchings of size  $k$  and  $k - 1$  (the rapid mixing of this walk is shown in [8]). For bipartite graphs [17] and planar graphs [1], there are polynomial time algorithms to approximately sample from the uniform distribution on  $\mathcal{M}_k(G)$  for all  $1 \leq k \leq m^*(G)$ .

Perhaps most relevant to this note is recent work of Jain, Michelen, Pham, and Vuong [13] which established optimal mixing of the down-up walk on *independent sets* of a given size  $1 \leq k \leq (1 - \delta)\alpha_c(\Delta)n$ , for the class of  $n$  vertex graphs  $G$  with maximum degree  $\Delta$  (using the spectral independence framework). Here,  $\alpha_c(\Delta)$  is a function such that the problem of (approximately) sampling independent sets of size  $k > \alpha_c(\Delta)n$  on  $n$  vertex graphs with maximum degree  $\Delta$  is computationally intractable, unless  $\text{NP} = \text{RP}$ ; this was shown by Davies and Perkins [9]. In the same paper, Davies and Perkins showed that by combining the rapid mixing of the Glauber dynamics for the hard-core model in the tree uniqueness regime ([3, 6]) with a rejection sampling step, one can obtain a polynomial time algorithm to approximately sample from the uniform distribution on independent sets of size  $k$  provided that  $1 \leq k \leq (1 - \delta)\alpha_c(\Delta)n$ ; this is entirely analogous to how [16, 6] imply polynomial time approximate samplers for the uniform distribution on  $\mathcal{M}_k(G)$  for  $1 \leq k \leq (1 - \delta)m^*(G)$ . Davies and Perkins conjectured [9, Conjecture 5] that the down-up walk for independent sets mixes in polynomial time provided that  $1 \leq k \leq (1 - \delta)\alpha_c(\Delta)n$  and this was resolved (in a stronger form) by [13]; our work may be viewed as resolving the analog of the conjecture of Davies and Perkins for matchings.

Finally, we remark that there is a large body of literature in probability concerned with the mixing of analogous walks for product(-like) domains with conservation laws (in our setting, the size of the matching is a conserved quantity); see, e.g., [5, 12] and the references therein. In our setting, the base measure (the natural choice is the monomer-dimer model at a suitable activity) is significantly more complicated and very far from being a product distribution, although we remark works are often able to exploit product structure and other symmetries to obtain rather precise results.

## 1.2 Organization

In Section 2, we present the proof of Theorem 1. In Section 3, we discuss barriers to a potential spectral independence approach for proving Theorem 1. In each (sub)section, we begin with an overview of the proof and some motivation.

## 2 Proof of Theorem 1

### 2.1 Preliminaries

Let  $P$  denote the transition matrix of an ergodic Markov chain on the finite state space  $\Omega$ , which is reversible with respect to the (unique) stationary distribution  $\pi$ . Let  $E(P) = \{(x, y) \in \Omega \times \Omega : P(x, y) > 0\}$  denote the “edges” of the transition matrix. Recall that the *Dirichlet form* is defined for  $f, g : \Omega \rightarrow \mathbb{R}$  by

$$\mathcal{E}_P(f, g) := \frac{1}{2} \sum_{x, y \in \Omega} \pi(x) P(x, y) (f(x) - f(y))(g(x) - g(y)).$$

The *spectral gap*  $\alpha$  is defined to be the largest value such that for all  $\varphi : \Omega \rightarrow \mathbb{R}$ ,

$$\alpha \operatorname{Var}_\pi[\varphi] \leq \mathcal{E}_P(\varphi, \varphi).$$

The (total-variation) *mixing time* is defined by

$$\tau_{\text{mix}} = \max_{x \in \Omega} \min\{t : d(P^t x, \pi)_{\text{TV}} < 1/4\},$$

where  $d(\cdot, \cdot)_{\text{TV}}$  denotes the total variation distance between probability distributions.<sup>4</sup> The following relationship between the spectral gap and the mixing time is standard (see, e.g. [20]):

$$\tau_{\text{mix}} \leq \alpha^{-1} \log\left(\frac{1}{\min_{x \in \Omega} \pi(x)}\right). \quad (1)$$

In order to bound the spectral gap of the down-up walk, we will use the technology of multicommodity flows ([22, 11]).

► **Definition 3.** Consider the undirected graph  $H = (\Omega, E(P))$ . For  $x, y \in \Omega$ , let  $\mathcal{Q}_{xy}$  denote the set of all simple paths from  $x$  to  $y$  in  $H$ . Let  $\mathcal{Q} = \bigcup_{x, y} \mathcal{Q}_{xy}$ . A *flow* is a function  $f : \mathcal{Q} \rightarrow \mathbb{R}_{\geq 0}$  such that  $\sum_{q \in \mathcal{Q}_{xy}} f(q) = \pi(x)\pi(y)$ . Given a flow  $f$ , we define its cost by

$$\rho(f) = \max_{(x, y) \in E(P)} \frac{1}{\pi(x)P(x, y)} \sum_{q \ni (x, y)} f(q),$$

and its length by

$$\ell(f) = \max_{q: f(q) > 0} |q|,$$

where  $|q|$  denotes the number of edges in the path  $q$ .

It was shown in [22, 11] that any flow gives a lower bound on the spectral gap.

► **Theorem 4.** Let  $P$  be a reversible ergodic Markov chain and  $f$  be a flow. Then the spectral gap  $\alpha$  satisfies

$$\alpha \geq 1/(\rho(f)\ell(f)).$$

For each  $t \in E(P)$ , let  $\text{paths}(t) = \{q \ni t : f(q) > 0\}$ . A common tool (see, e.g., [22]) for bounding  $\rho(f)$  is a *flow encoding*, which is a collection of maps  $\eta_t : \text{paths}(t) \rightarrow \Omega$  for all  $t \in E(P)$ . If all the maps are poly( $n$ )-to-one and the measure  $\pi$  is “fairly tame” (the uniform measure on  $\Omega$  automatically satisfies this condition), then this gives an inverse polynomial bound on the spectral gap.

<sup>4</sup> Note that the quantity  $1/4$  is fairly arbitrary here. Replacing  $1/4$  with  $\varepsilon$  increases the mixing time by at most a factor of  $\log_2(\varepsilon^{-1})$ .



## 2.2 Constructing a flow

Recall that  $1 \leq k \leq (1 - \delta)m^*(G)$  and  $\Omega$  denotes the set of matchings in  $G$  of size exactly  $k$ . Given two matchings  $x, y \in \Omega$ , our flow will be constructed by uniformly distributing the demand  $\pi(x)\pi(y)$  over a collection of carefully constructed paths. Compared to the construction of a flow in [16], we face two challenges:

- First, since we are not working with perfect matchings (or matchings which are a constant additive size away from being perfect), using just one path to route all the flow for each pair of matchings in the natural fashion results in a flow with exponentially high cost. To get around this issue, we use the (standard) idea of distributing the flow uniformly among essentially all possible paths, as is done for the Bernoulli-Laplace model (see [22]) and also for a random walk on the union of matchings of size  $k$  and  $k - 1$  [8].
- Second – and this is the main new ingredient in our construction – our state space consists of matchings of a fixed size, whereas all previous walks and flow constructions (e.g. [16, 8]) required working with matchings of at least two adjacent sizes. In order to route flow along such paths while still incurring only polynomial cost, we divide pairs of matchings into a “good” set and a “bad” set depending on the combinatorial structure of the symmetric difference. For the good set, it is fairly simple to construct a flow, incorporating the above idea of distributing the flow uniformly among all possible paths. For a pair in the bad set  $(x, y)$ , we show that there is a nearby good pair  $(\tilde{x}, y)$ , in the sense that  $x$  can be transformed into  $\tilde{x}$  using a short path. The fact that we can transform  $x$  to a suitable  $\tilde{x}$  with a short path is key to bounding the cost of the flow, and this is where we use that  $k \leq (1 - \delta)m^*(G)$ .

Let  $x \oplus y$  denote the symmetric difference  $(x \setminus y) \cup (y \setminus x)$ . Since  $x$  and  $y$  are each matchings and so have maximum degree 1,  $x \oplus y$  is a disjoint union of paths and even-length cycles. For the sake of analysis, place arbitrary total orders on the set of even length paths in  $G$  and the set of even length cycles in  $G$ . Associate to each cycle one arbitrary distinguished vertex and to each odd-length path one arbitrary distinguished endpoint. These will all remain fixed for the remainder of the paper.

We partition  $\Omega^2 := \Omega \times \Omega$  into  $(\Omega^2)_g \cup (\Omega^2)_b$  where

$$(\Omega^2)_b = \{(x, y) : x \oplus y \text{ contains a cycle and no odd-length paths}\},$$

$$(\Omega^2)_g = \Omega^2 \setminus (\Omega^2)_b.$$

We will first describe the collection of paths between the “good pairs”  $(\Omega^2)_g$ . Later, we will leverage this collection on paths along with an additional idea to obtain a suitable collection of paths between the “bad pairs”  $(\Omega^2)_b$ .

### Good pairs

Let  $(x, y) \in (\Omega^2)_g$ . The symmetric difference  $x \oplus y$  consists of even length paths, even length cycles, odd length paths with more edges in  $y$  (which are necessarily  $x$ -augmenting paths), and odd length paths with more edges in  $x$  (which are necessarily  $y$ -augmenting paths). We have an induced ordering on the even paths and the cycles from our total order. Since  $|x| = |y|$ ,  $x \oplus y$  contains the same number of  $x$ -augmenting and  $y$ -augmenting paths; suppose there are  $2j$  total odd-length paths. Let  $\sigma_x, \sigma_y$  be permutations of the sets of  $x$ -augmenting,  $y$ -augmenting paths respectively. For each such choice of  $(\sigma_x, \sigma_y)$ , we construct a path as follows from  $x$  to  $y$  in  $\Omega$ . Before proceeding to the formal details, let us briefly describe the procedure: we first change  $x$  to  $y$  along all even paths. We then change  $x$  to  $y$  along the

first  $x$ -augmenting path  $\sigma_x(1)$ ; at the end of such a path, there is an additional  $y$ -edge to be added, which gives us the necessary room to switch from  $x$  to  $y$  along all cycles, while still remaining in  $\Omega$ . At the end of the cycle processing stage, there is an additional  $y$ -edge to be added; we pair this up with switching  $x$  to  $y$  along the first  $y$ -augmenting path  $\sigma_y(1)$ . Finally, we switch  $x$  to  $y$  along pairs of  $x$ -augmenting and  $y$ -augmenting paths  $\sigma_x(i), \sigma_y(i)$  in the natural fashion.

Formally, set  $M_0 = x$  and proceed as follows:

1. Process all even length paths in order. To process an even path, enumerate the edges  $e_1, e_2, \dots, e_{2\ell}$  such that  $e_i \cap e_{i+1} \neq \emptyset$  and  $e_1 \in y$ . This places all odd edges in  $y$  and the evens in  $x$ . Suppose  $t$  steps have been taken. First make the transition  $M_{t+1} = M_t \cup e_1 \setminus e_2$ ,<sup>5</sup> then  $M_{t+2} = M_{t+1} \cup e_3 \setminus e_4$ , and continue until  $M_{t+\ell} = M_{t+\ell-1} \cup e_{2\ell-1} \setminus e_{2\ell}$ . After processing all even paths, if we have reached  $y$ , terminate.
2. Process the first  $x$ -augmenting path  $p = \sigma_x(1)$ . Let  $e^* \in p$  be the edge incident to the distinguished endpoint. Process  $p \setminus e^*$  as an even path as in step (1), leaving only  $e^*$  to be added.
3. Process all cycles in order. For a cycle  $c$ , let  $e$  (respectively  $e'$ ) be the edge in  $c \cap x$  (respectively  $c \cap y$ ) incident to the distinguished vertex of  $c$ . First, let  $M_{t+1} = M_t \cup e^* \setminus e$  to complete the previous path and puncture the cycle. Now, process  $c \setminus \{e, e'\}$  as an even path as in step (1). Label  $e^* := e'$  and process the next cycle in the same way. At the end of this step, some  $e^*$  will remain.
4. Process the first  $y$ -augmenting path  $p = \sigma_y(1)$ . Let  $e \in p$  be the edge incident to the distinguished endpoint. Begin with  $M_{t+1} = M_t \cup e^* \setminus e$ , then process  $p \setminus e$  as an even path as in step (1).
5. Process any remaining  $x$ -augmenting and  $y$ -augmenting paths in pairs  $p = \sigma_x(i), p' = \sigma_y(i)$ . Let  $e$  (respectively  $e'$ ) denote the edges incident to the distinguished endpoints of  $p$  (respectively  $p'$ ). First process  $p \setminus e$  as an even path, then exchange  $M_{t+1} = M_t \cup e \setminus e'$ , then process  $p' \setminus e'$  as an even path.

This defines a unique path from  $x$  to  $y$  for any two permutations  $\sigma_x, \sigma_y$ , and so gives  $(j!)^2$  total paths  $x \rightarrow y$ . We uniformly distribute the demand  $\pi(x)\pi(y) = 1/|\Omega^2|$  by setting  $f(q) = 1/(|\Omega|^2(j!)^2)$  for each path  $q$  thus defined.

### Bad pairs

Given  $(x, y) \in (\Omega^2)_b$ , we will route the flow through  $(\Omega^2)_g$  by choosing some suitable  $(\tilde{x}, y) \in (\Omega^2)_g$  and adding a suitable prefix to all paths (as above) from  $\tilde{x}$  to  $y$ . Since  $|x| \leq (1 - \delta)m^*(G)$ ,  $x$  has some augmenting path  $p$  of length at most  $2\delta^{-1}$ . This follows from the pigeonhole principle: for  $M^*$  a maximum matching in  $G$ ,  $x \oplus M^*$  is a graph with at most  $2m^*(G)$  non-isolated vertices and at least  $\delta m^*(G)$  disjoint  $x$ -augmenting paths, so that there must be an  $x$ -augmenting path of length at most  $2\delta^{-1}$ . Consider now  $x^+ := x \oplus p$ ; this is a matching of size  $k + 1$ . We claim that there exists some  $e \in x^+$  such that for  $\tilde{x} := x^+ \setminus e$  satisfies  $(\tilde{x}, y) \in (\Omega^2)_g$ . To see this, note that since  $|x^+| > |y|$ ,  $x^+ \oplus y$  contains a  $y$ -augmenting path  $p'$ . If  $x^+ \subset p'$ , then  $x^+ \oplus y$  cannot contain a cycle, as  $p'$  is an alternating path between edges in  $x^+$  and in  $y$ , and  $|x^+| = |y| + 1$ , and so  $y \subset p'$  as well. Thus  $x^+ \oplus y = p'$  is a single path and contains no cycles, so we may choose any  $e \in x^+$  and  $(\tilde{x}, y) \in (\Omega^2)_g$ . Otherwise, by choosing any edge  $e \in x^+ \setminus p'$ , we guarantee  $\tilde{x} \oplus y$  has odd-length paths (in particular,  $p'$ ) and so  $(\tilde{x}, y) \in (\Omega^2)_g$ .

<sup>5</sup> As a slight abuse of notation, when  $m$  is a matching, we will write  $m \cup e$  to mean  $m \cup \{e\}$  and similarly  $m \setminus e$  instead of  $m \setminus \{e\}$ .

For every pair  $(x, y) \in (\Omega^2)_b$ , we make a fixed (but otherwise arbitrary) choice of  $p$  (an  $x$ -augmenting path of length at most  $2\delta^{-1}$ ) and  $e$  as above. For a path  $\tilde{q} \in \mathcal{Q}_{\tilde{x}y}$ , we define  $q \in \mathcal{Q}_{xy}$  as follows.

1. Process  $p$  as an  $x$ -augmenting path (see previous step 2), leaving some  $e^*$  to be added.
2. Make the exchange  $M_{t+1} = M_t \cup e^* \setminus e$ , arriving at  $\tilde{x}$ .
3. Follow the path  $\tilde{q}$ .

We assign  $f(q) = f(\tilde{q})$  so that the same amount of flow is routed from  $x$  to  $y$  as from  $\tilde{x}$  to  $y$ . We remark that choosing an augmenting path of length  $O_\delta(1)$  is crucial for bounding the cost of the flow below.

### 2.3 Flow encoding

For  $t \in E(P)$ , we now bound  $f(t) := \sum_{q \ni t} f(q)$  using the method of flow encodings. Recall that  $\text{paths}(t) = \{q \ni t : f(q) > 0\}$ . Fix some transition  $t = (z, z') \in E(P)$ . We will partition  $\text{paths}(t)$  into three sets and bound the contribution to  $f(t)$  from each of the three using a ‘‘partial flow encoding’’. We have the ‘‘good’’ paths  $\text{paths}_g(t)$  consisting of paths  $q \in \text{paths}(t)$  whose endpoints are in  $(\Omega^2)_g$ . Recall that the paths in  $(\Omega^2)_b$  consist of two phases: the prefix from  $x \rightarrow \tilde{x}$ , and then a good path from  $\tilde{x} \rightarrow y$ . Denote by  $\text{paths}_a(t)$  those paths which use the transition  $t$  in the prefix  $x \rightarrow \tilde{x}$ , and by  $\text{paths}_b(t)$  those paths which use the transition  $t$  in the path from  $\tilde{x} \rightarrow y$ . We will frequently need the set of short paths in  $G$

$$\mathcal{P}_\delta := \left\{ (v_1 v_2 \cdots v_\ell) : \{v_i, v_{i+1}\} \in E(G), \ell \leq 2/\delta \right\}.$$

We will construct  $\Omega \times \mathcal{P}_\delta$ -valued functions  $\eta_g, \eta_b, \eta_a$  on these subsets of  $\text{paths}(t)$ .

#### Construction of $\eta_g$

We first construct  $\eta_g : \text{paths}_g(t) \rightarrow \Omega \times \mathcal{P}_\delta$ . Let  $t = (z, z') \in E(P)$ . For a path  $q$ , let  $q^-, q^+$  be the endpoints. Let  $m = q^- \oplus q^+ \oplus (z \cup z')$ . It is easily checked that  $m$  is a matching of size  $k - 1$  and that  $m' \oplus (z \cup z') = q^- \oplus q^+$  (the same construction is used in [16]). For consistency, we further map  $m$  into an element of  $\Omega \times \mathcal{P}_\delta$  using a fixed (but otherwise arbitrary)  $m'$ -augmenting path  $p \in \mathcal{P}_\delta$ . The existence of a short  $m$ -augmenting path is guaranteed by the fact that  $|m| < (1 - \delta)m^*(G)$ . Formally, we have

$$\begin{aligned} \eta_g : \text{paths}_g(t) &\rightarrow \Omega \times \mathcal{P}_\delta \\ q &\mapsto (q^- \oplus q^+ \oplus (z \cup z') \oplus p, p). \end{aligned}$$

Note that given the image  $(m \oplus p, p)$ , we take  $(m' \oplus p) \oplus p$  to recover  $m$ , which then recovers  $q^- \oplus q^+$  as before. We can now reindex the sum

$$\sum_{q \in \text{paths}_g(t)} f(q) = \sum_{(m,p) \in \Omega \times \mathcal{P}_\delta} \sum_{q \in \eta_g^{-1}(m,p)} f(q).$$

The endpoints of every  $q \in \eta_g^{-1}(m, p)$  have the same symmetric difference as noted above. Let this symmetric difference have  $2j$  odd-length paths. By our construction of the flow,  $f(q) = |\Omega|^{-2} (j!)^{-2}$  for all  $q \in \eta_g^{-1}(m, p)$ . We now count how many paths use the transition  $(z, z')$  based on which  $G$ -paths it is processing. This requires some case analysis, but ultimately, is based on blending the analysis of the flow encoding for the Glauber dynamics for the monomer-dimer model in [16] with the flow encoding for the Bernoulli-Laplace model [22].

- Case I:** If  $(z, z')$  is processing even length paths, then we can use our total order on even length paths to identify which parts of each cycle belong to  $q^-$  and to  $q^+$ . We know that we have not yet begun processing odd paths or cycles, so the parts in  $z$  belong to  $q^-$  and those outside  $z$  belong to  $q^+$ . We thus know  $q^-$  and  $q^+$  and so there are exactly  $(j!)^2$  paths using  $(z, z')$ .
- Case II:** If  $(z, z')$  finishes processing an odd-length  $G$ -path and begins processing a cycle, then we know the odd  $G$ -path is the first such processed, and by the same reasoning as before, we can deduce the endpoints  $q^-$  and  $q^+$ . We also know  $\sigma_x(1)$  is the path intersecting  $z \oplus z'$ . The remainder of  $\sigma_x$  and the entirety of  $\sigma_y$  is free, so there are  $(j!)^2/j$  paths using  $(z, z')$ .
- Case III:** If  $(z, z')$  is processing entirely cycles, then  $z$  has a perfect matching on  $j + 1$  of the odd-length paths and the interior edges on  $j - 1$  of the odd-length paths. One of these  $G$ -paths has already been augmented, and then there will be a path for every ordering of the remaining  $G$ -paths. Thus there are  $(j + 1)j!(j - 1)! = (1 + 1/j)(j!)^2$  paths.
- Case IV:** If  $(z, z')$  finishes a cycle and begins an odd-length path, then  $z$  has a perfect matching on  $j + 1$  odd paths (one of which is being de-augmented in  $(z, z')$ ) and the interior edges on  $j - 1$  odd paths. The path touched by  $z \oplus z'$  is  $\sigma_y(1)$ . We must choose one of the remaining  $j$  perfectly matched paths to be  $\sigma_x(1)$ , and then the remainder of  $\sigma_x, \sigma_y$  are free on the sets of  $j - 1$  interior, perfect paths respectively. There are thus  $j((j - 1)!)^2 = (j!)^2/j$  paths using  $(z, z')$ .
- Case V:** If  $(z, z')$  is augmenting an odd-length path, then of the other  $2j - 1$  paths,  $z$  is perfect on  $j$  and interior on  $j - 1$ . Suppose  $2r$  paths have already been processed. Then we may choose the already-augmented paths ( $\binom{j}{r}$  choices), the already-de-augmented paths ( $\binom{j-1}{r}$  choices), the order for each ( $(r!)^2$  choices), and the order for the remaining augmentations and de-augmentations ( $(j - r)!(j - 1 - r)!$  choices). This gives  $j!(j - 1)!$  paths through  $(z, z')$  that have already processed  $r$  pairs of  $G$ -paths. We now sum over  $0 \leq r \leq j - 1$  to get  $j(j!)(j - 1)! = (j!)^2$  total paths through  $(z, z')$ .
- Case VI:** If  $(z, z')$  is de-augmenting an odd-length path, then by the same logic but with  $j - 1$  perfect paths and  $j$  interior paths, we again have  $(j!)^2$  total paths.
- Case VII:** If  $(z, z')$  finishes augmenting one odd path and begins de-augmenting the next, then we know these  $G$ -paths occur adjacently in the path  $q$ . Then by the same reasoning as Case V but with  $j - 1$  perfect paths and  $j - 1$  interior paths, we will get  $(j!)^2/j$  total paths through  $(z, z')$ .

In all cases, we have at most  $(1 + 1/j)(j!)^2 \leq 2(j!)^2$  paths in  $\eta_g^{-1}(m, p)$  that use the transition  $(z, z')$ . As each has weight  $f(q) = |\Omega|^{-2}(j!)^{-2}$ , this means the inner sum is at most  $2/|\Omega|^2$  and so we can bound

$$\sum_{q \in \text{paths}_g(t)} f(q) = \sum_{(m,p) \in \Omega \times \mathcal{P}_\delta} \sum_{q \in \eta_g^{-1}(m,p)} f(q) \leq \sum_{(m,p) \in \Omega \times \mathcal{P}_\delta} \frac{2}{|\Omega|^2} = \frac{2|\mathcal{P}_\delta|}{|\Omega|}. \quad (2)$$

### Construction of $\eta_b$

Recall that  $\text{paths}_b(t)$  are those paths that use the transition  $t$  as part of following a “good” path from  $\tilde{x}$  to  $y$ . Thus we may instead choose the good path  $q$  that is routed through  $t$ , and then count how many starting points  $x$  could route through  $q^-$  (the starting point of  $q$ ) to get to  $q^+$  (the ending point of  $q$ ). By the construction of our paths, this requires  $x \oplus q^-$  to be a single short augmenting  $G$ -path in  $\mathcal{P}_\delta$  together with a single edge in  $E(G)$ . These

together will uniquely determine the total path  $x \rightarrow q^+$ . Thus each good path through  $t$  is used in at most  $|\mathcal{P}_\delta| |E(G)|$  bad paths, and the value of  $f$  is unchanged by the prefix, so we may bound using Equation (2)

$$\sum_{q \in \text{paths}_b(t)} f(q) \leq \sum_{\tilde{q} \in \text{paths}_g(t)} |\mathcal{P}_\delta| |E(G)| f(\tilde{q}) \leq \frac{2|\mathcal{P}_\delta|^2 |E(G)|}{|\Omega|}. \quad (3)$$

### Construction of $\eta_a$

Finally, for  $t = (z, z^-)$  and  $q \in \text{paths}_a(t)$ , let  $p$  be the  $G$ -path that is augmented during the prefix. Then, define the function

$$\begin{aligned} \eta_a : \text{paths}_a(t) &\rightarrow \Omega \times \mathcal{P}_\delta \\ q &\mapsto (q^+, p). \end{aligned}$$

Suppose  $q \in \eta_a^{-1}(m, p)$  for some matching  $m \in \Omega$  and  $G$ -path  $p \in \mathcal{P}_\delta$  and let  $t = (z, z^-)$ . Then we know that  $q^+ = m$ , and we know that  $q^-$  consists of  $z \setminus p$  and the interior alternating edges of  $p$ . Thus all paths in  $\eta_a^{-1}(m, p)$  have the same endpoints, and so their total flow is at most the net flow between those two points, which is  $|\Omega|^{-2}$ . We can then calculate

$$\sum_{q \in \text{paths}_a(t)} f(q) = \sum_{(m,p) \in \Omega \times \mathcal{P}_\delta} \sum_{q \in \eta_a^{-1}(m,p)} f(q) \leq \sum_{(m,p) \in \Omega \times \mathcal{P}_\delta} \frac{1}{|\Omega|^2} = \frac{|\mathcal{P}_\delta|}{|\Omega|}. \quad (4)$$

### Bounding the cost of the flow

Using Equation (2), Equation (3), and Equation (4), for any transition  $t \in E(P)$ ,

$$\sum_{q \in \text{paths}(t)} f(q) = \sum_{q \in \text{paths}_g(t)} f(q) + \sum_{q \in \text{paths}_b(t)} f(q) + \sum_{q \in \text{paths}_a(t)} f(q) \leq \frac{3|\mathcal{P}_\delta| + 2|\mathcal{P}_\delta|^2 |E(G)|}{|\Omega|}. \quad (5)$$

Since  $\pi(z) = 1/|\Omega|$  for all  $z \in \Omega$  and  $P(z, z') \geq 1/(k|E(G)|)$  (since the possible transitions from  $z$  consist of removing one of  $k$  edges and adding one of  $|E(G)|$  edges), we get that

$$\rho(f) \leq |\Omega| \cdot k|E(G)| \cdot \frac{3|\mathcal{P}_\delta| + 2|\mathcal{P}_\delta|^2 |E(G)|}{|\Omega|} \leq 3k|E(G)|^2 |\mathcal{P}_\delta|^2.$$

Finally, let  $\Delta$  denote the maximum degree of  $G$  and note that  $|\mathcal{P}_\delta| \leq 2n\Delta^{2/\delta-1}$  to get that

$$\rho(f) \leq 12k|E(G)|^2 \cdot n^2 \Delta^{4/\delta-2}. \quad (6)$$

## 2.4 Rapid mixing

We will use Theorem 4 to bound the spectral gap via the flow  $f$  defined in Section 2.2. Note that the down-up walk is reversible with respect to the uniform distribution, aperiodic since  $P(x, x) > 0$ , and irreducible (for instance, by using the paths used in our flow  $f$ ). Therefore, the assumptions of Theorem 4 are satisfied. To bound the maximum length  $\ell(f)$  of any path used in our flow, note that by construction, any edge in  $G$  is included in at most three exchanges. Hence,  $\ell(f) \leq 3|E(G)|$ . Combining this with Equation (6), we see that the spectral gap  $\alpha$  of the down-up walk satisfies

$$\alpha^{-1} \leq 36k|E(G)|^3 \cdot n^2 \Delta^{4/\delta-2}. \quad (7)$$

Finally, the mixing time bound in Theorem 1 follows from Equation (1) by noting that  $\log |\Omega| \leq \log 2^{|E(G)|} \leq |E(G)|$ .

### 3 Barriers to the spectral independence approach

For a distribution  $\pi$  on  $\binom{[n]}{k}$ , we define the (signed) pairwise influence matrix  $M_\pi \in \mathbb{R}^{n \times n}$  by

$$M_\pi(i, j) = \begin{cases} 0 & \text{if } j = i, \\ \mathbb{P}_\pi[j | i] - \mathbb{P}_\pi[j | \bar{i}] & \text{otherwise.} \end{cases}$$

where  $\mathbb{P}_\pi[i] = \mathbb{P}_{S \sim \pi}[i \in S]$  and  $\mathbb{P}_\pi[\bar{i}] = \mathbb{P}_{S \sim \pi}[i \notin S]$ . We say that  $\pi$  is  $\eta$ -spectrally independent (at link  $\emptyset$ ) if  $\lambda_{\max}(M_\pi) \leq \eta$  and that  $\pi$  is  $\eta$ - $\ell_\infty$ -independent (at link  $\emptyset$ ) if  $\max_{i \in [n]} \sum_{j=1}^n |M_\pi(i, j)| \leq \eta$ . Note that the latter condition implies the former.

We begin by noting that for the class of bounded degree graphs, for  $k$  bounded away from the matching number, the uniform distribution on matchings of size  $k$  is  $O(1)$ - $\ell_\infty$ -independent.

► **Proposition 5.** *Let  $G = (V, E)$  be a graph on  $n$  vertices with maximum degree  $\Delta$ . Let  $\delta > 0$  and for  $1 \leq k \leq (1 - \delta)m^*(G)$ , let  $\pi$  be the uniform distribution on matchings of  $G$  of size  $k$ . Then  $\pi$  is  $O_{\delta, \Delta}(1)$ - $\ell_\infty$ -independent (at link  $\emptyset$ ).*

**Proof.** For  $k = o(n)$ , this is implied by a coupling argument (e.g. [21]). For  $k = \Omega(n)$ , the proof follows from the same argument as in the proof of [13, Theorem 8]: the differences are that we compare to the monomer-dimer model at activity  $\lambda = O_{\delta, \Delta}(1)$  using [14, Lemma 4.1], replace [13, Theorem 9] by [6, Theorem 2.10], and replace [13, Theorem 15] by a suitable multivariate zero-free region for the matching polynomial (e.g. [7]). We omit further details. ◀

Given Proposition 5, one might hope to obtain an inverse polynomial bound on the spectral gap of the down-up walk (at least for the class of bounded degree graphs) using the powerful spectral independence framework as is done, for instance, in the case of the down-up walk on independent sets of a fixed size in [13]; we refer the reader to [3] for an introduction to this framework. In order to do this, we need to show that the distribution remains  $O_{\delta, \Delta}(1)$ -spectrally independent under any pinning. In our situation, a pinning  $\tau$  is a matching of size  $\ell < k$ . We would then consider  $\Omega_\tau = \{m \in \Omega : \tau \subset m\}$  under the distribution induced by  $\pi$  (uniform, in our case), and show  $O_{\delta, \Delta}(1)$ -spectral independence of this space. We note that there is a more powerful “average-case” version of this argument, which (roughly) allows us to consider typical pinning obtained by starting from some fixed matching of size  $k$  and pinning a random subset of  $k - \ell$  edges to be included (see [4, 2]). We present barriers to this approach.

- We observe that such an approach cannot work for the down-up walk. Indeed, if it were to work, then one would also be able to show that the down-up walk has inverse polynomial spectral gap for the induced uniform distribution on size  $\ell$  matchings obtained by starting with an arbitrary matching of size  $k$  and pinning a uniform subset of  $k - \ell$  edges to belong to the matching. However, as we discuss below, it is easy to construct an example where even for polynomially large  $\ell$ , with high probability, the down-up walk is not even ergodic (Claim 7).
- In the above example, the failure of ergodicity may be circumvented by using an  $O(1)$ -step down-up walk. However, it is still the case that proving mixing of the  $O(1)$ -step down-up walk using the (average) spectral independence framework necessitates proving mixing for the  $O(1)$ -step down-up walk for the aforementioned induced distributions on matchings of size  $\ell$ . We present a construction (Claim 6) showing that these induced distributions can correspond to the uniform distribution on size  $\ell$  matchings in pretty arbitrary graphs

with matching number  $\ell(1 + o(1))$ ; hence, there does not seem to be a way to use this machinery without basically showing that  $O(1)$ -step down-up walks mix rapidly for (almost) maximum matchings in arbitrary bounded-degree graphs, which is a major open problem.

Our examples will follow the same general template. To set up some notation, given a graph  $G = (V, E)$  and a pinning  $\tau$  (a matching  $\tau$  in  $G$ ), define the *residual graph*  $G_\tau$  to be the induced subgraph

$$G_\tau = G[V(G_\tau)],$$

where

$$V(G_\tau) = V(G) \setminus \bigcup_{e \in \tau} e.$$

Sampling from the uniform distribution on matchings in  $G$  of size  $k$ , conditioned on pinning  $\tau$  to be in the matching, is equivalent to finding a matching of size  $k - |\tau|$  in  $G_\tau$ .

We are now ready to construct our examples. Fix some  $0 < \delta < 1/5$  the desired gap from maximality, as in the statement of Theorem 1. We define a graph  $G = (V, E)$  where  $|V| = n$  as follows:  $G$  consists of  $\delta n/2$  disjoint copies of  $P_9$  (the path with 10 vertices and 9 edges) and an arbitrary graph  $G'$  on the remaining  $(1 - 5\delta)n$  vertices such that  $G'$  has a perfect matching. Let  $M$  be the matching given by taking the union of a perfect matching  $M'$  in  $G'$  with the interior alternating edges on each  $P_9$ ; note that  $|M| = n/2 - \delta n/2 = (1 - \delta)m^*(G)$ . We will be considering pinning a uniform random subset of  $M$  of a fixed size.

▷ **Claim 6.** For a random pinning  $\tau$  of size  $(1 - \lambda)|M|$ , the ratio

$$\mathbb{E} \left[ 1 - \frac{|M| - |\tau|}{m^*(G_\tau)} \right] = O(\delta\lambda^4).$$

The implication of this claim is that, while we started with the uniform distribution on matchings of size at most  $(1 - \delta)$  of the maximum matching in  $G$ , we now need to deal with the uniform distribution on matchings of size at least  $(1 - \delta\lambda^4)$  times the maximum matching in  $G_\tau$ .

*Proof.* Let  $\tau$  be a random pinning of size  $(1 - \lambda)|M|$ . Let  $X_\tau$  be the number of  $P_9$ s that  $\tau$  does not intersect. Then by linearity of expectation,

$$\mathbb{E}[X_\tau] = \frac{\delta n}{2} \Pr[\tau \text{ avoids a fixed } P_9] = O(\delta n \lambda^5). \tag{8}$$

The key observation here is that once we have pinned any edge in  $M \cap P_9$  for some copy of  $P_9$ , we have split  $P_9$  into two even paths and are demanding a maximum matching on each of those. Hence, by construction, we see that  $m^*(G_\tau) = |M| - |\tau| + X_\tau$ . We now compute

$$\begin{aligned} \mathbb{E} \left[ 1 - \frac{|M| - |\tau|}{m^*(G_\tau)} \right] &= 1 - \mathbb{E} \left[ \frac{|M| - |\tau|}{|M| - |\tau| + X_\tau} \right] \leq 1 - \frac{|M| - |\tau|}{|M| - |\tau| + \mathbb{E}[X_\tau]} \\ &\leq \frac{\mathbb{E}[X_\tau]}{\lambda|M|} = O(\delta\lambda^4), \end{aligned}$$

where the first line uses Jensen's inequality and the second line uses Equation (8). ◀

In the above construction, take  $G'$  to be a disjoint union of  $(1 - 5\delta)n/4$  copies of  $C_4$  (the 4-cycle) and accordingly, take  $M'$  to be a union of perfect matchings on each  $C_4$ .

## 63:12 Rapid Mixing of the Down-Up Walk on Matchings of a Fixed Size

▷ Claim 7. For a random pinning  $\tau$  of size  $n/2 - n^{2/3}$ , with high-probability, the down-up walk on matchings of size  $|M| - \tau$  on the induced graph  $G_\tau$  is not ergodic.

Proof. It suffices to show that for a random pinning  $\tau$  of size  $n/2 - n^{2/3}$ , with high probability, (i)  $\tau$  intersects every  $P_9$ , (ii)  $\tau$  fails to intersect some  $C_4$ .

For (i), by a union bound and direct computation, we get that

$$\mathbb{P}[\tau \text{ avoids some } P_9 \cap M] \leq \frac{\delta n}{2} \mathbb{P}[\tau \text{ avoids a fixed } P_9 \cap M] = O(\delta n^{-1/3}).$$

For (ii), we get that

$$\begin{aligned} \mathbb{P}[\tau \text{ intersects all } C_4] &= \mathbb{P}[\tau^c \text{ contains no } C_4 \cap M] \\ &\leq n^{O(1)} \mathbb{P}[\tau^c \text{ does not contain a fixed } C_4 \cap M]^{(1-5\delta)n/4} \\ &\leq \exp(-\Theta(n^{1/3})), \end{aligned}$$

where the second follows by comparing probabilities between the independent model of density  $\Theta(n^{-1/3})$  and the slice model and the last line follows by direct computation. The union bound now shows that with high probability, (i) and (ii) simultaneously hold. ◁

---

### References

- 1 Yeganeh Alimohammadi, Nima Anari, Kirankumar Shiragur, and Thuy-Duong Vuong. Fractionally log-concave and sector-stable polynomials: counting planar matchings and more. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 433–446, 2021.
- 2 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Universality of spectral independence with applications to fast mixing in spin glasses. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5029–5056. SIAM, 2024.
- 3 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *IEEE Symposium on Foundations of Computer Science*, 2020.
- 4 Nima Anari, Yang P Liu, and Thuy-Duong Vuong. Optimal sublinear sampling of spanning trees and determinantal point processes via average-case entropic independence. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 123–134. IEEE, 2022.
- 5 Pietro Caputo. Spectral gap inequalities in product spaces with conservation laws. In *Stochastic analysis on large scale interacting systems*, volume 39, pages 53–89. Mathematical Society of Japan, 2004.
- 6 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*, pages 1537–1550, 2021.
- 7 Zongchen Chen, Kuikui Liu, and Eric Vigoda. Spectral independence via stability and applications to holant-type problems. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 149–160. IEEE, 2022.
- 8 Paul Dagum, Michael Luby, Milena Mihail, and U Vazirani. Polytopes, permanents and graphs with large factors. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 412–421. IEEE Computer Society, 1988.
- 9 Ewan Davies and Will Perkins. Approximately counting independent sets of a given size in bounded-degree graphs. In *48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198, pages 62:1–62:18, 2021.



- 10 Persi Diaconis and Mehrdad Shahshahani. Time to reach stationarity in the bernoulli–laplace diffusion model. *SIAM Journal on Mathematical Analysis*, 18(1):208–218, 1987.
- 11 Persi Diaconis and Daniel Stroock. Geometric bounds for eigenvalues of markov chains. *The annals of applied probability*, pages 36–61, 1991.
- 12 Yuval Filmus, Ryan O’Donnell, and Xinyu Wu. Log-sobolev inequality for the multislice, with applications. *Electronic Journal of Probability*, 27:1–30, 2022.
- 13 Vishesh Jain, Marcus Michelen, Huy Tuan Pham, and Thuy-Duong Vuong. Optimal mixing of the down-up walk on independent sets of a given size. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1665–1681. IEEE, 2023.
- 14 Vishesh Jain, Will Perkins, Ashwin Sah, and Mehtaab Sawhney. Approximate counting and sampling via local central limit theorems. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1473–1486, 2022.
- 15 Mark Jerrum. Two-dimensional monomer-dimer systems are computationally intractable. *Journal of Statistical Physics*, 48:121–134, 1987.
- 16 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.
- 17 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.
- 18 Pieter Kasteleyn. Graph theory and crystal physics. *Graph Theory and Theoretical Physics*, pages 43–110, 1967.
- 19 Tzong-Yow Lee and Horng-Tzer Yau. Logarithmic sobolev inequality for some models of random walks. *The Annals of Probability*, 26(4):1855–1873, 1998.
- 20 David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.
- 21 Kuikui Liu. From coupling to spectral independence and blackbox comparison with the down-up walk. *arXiv preprint*, 2021. [arXiv:2103.11609](https://arxiv.org/abs/2103.11609).
- 22 Alistair Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, probability and Computing*, 1(4):351–370, 1992.
- 23 Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.



# On the Communication Complexity of Finding a King in a Tournament

Nikhil S. Mande   

University of Liverpool, UK

Manaswi Paraashar   

University of Copenhagen, Denmark

Swagato Sanyal 

Indian Institute of Technology Kharagpur, India

Nitin Saurabh 

Indian Institute of Technology Hyderabad, India

---

## Abstract

A tournament is a complete directed graph. A source in a tournament is a vertex that has no in-neighbours (every other vertex is reachable from it via a path of length 1), and a king in a tournament is a vertex  $v$  such that every other vertex is reachable from  $v$  via a path of length at most 2. It is well known that every tournament has at least one king. In particular, a maximum out-degree vertex is a king. The tasks of finding a king and a maximum out-degree vertex in a tournament has been relatively well studied in the context of query complexity. We study the *communication complexity* of finding a king, of finding a maximum out-degree vertex, and of finding a source (if it exists) in a tournament, where the edges are partitioned between two players. The following are our main results for  $n$ -vertex tournaments:

- We show that the communication task of finding a source in a tournament is *equivalent* to the well-studied Clique vs. Independent Set (CIS) problem on undirected graphs. As a result, known bounds on the communication complexity of CIS [Yannakakis, JCSS'91, Göös, Pitassi, Watson, SICOMP'18] imply a bound of  $\tilde{\Theta}(\log^2 n)$  for finding a source (if it exists, or outputting that there is no source) in a tournament.
- The deterministic and randomized communication complexities of finding a king are  $\Theta(n)$ . The quantum communication complexity of finding a king is  $\tilde{\Theta}(\sqrt{n})$ .
- The deterministic, randomized, and quantum communication complexities of finding a maximum out-degree vertex are  $\Theta(n \log n)$ ,  $\tilde{\Theta}(n)$  and  $\tilde{\Theta}(\sqrt{n})$ , respectively.

Our upper bounds above hold for all partitions of edges, and the lower bounds for a specific partition of the edges.

One of our lower bounds uses a fooling-set based argument, and all our other lower bounds follow from carefully-constructed reductions from Set-Disjointness. An interesting point to note here is that while the deterministic query complexity of finding a king has been open for over two decades [Shen, Sheng, Wu, SICOMP'03], we are able to essentially resolve the complexity of this problem in a model (communication complexity) that is usually harder to analyze than query complexity.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity; Theory of computation  $\rightarrow$  Quantum complexity theory; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Communication complexity, tournaments, query complexity

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.64

**Category** RANDOM

**Related Version** *Full Version URL:* <https://arxiv.org/abs/2402.14751> [36]

**Funding** *Manaswi Paraashar:* M.P. is supported by ERC grant (QInteract, Grant Agreement No 101078107).



© Nikhil S. Mande, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 64; pp. 64:1–64:23



Leibniz International Proceedings in Informatics

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

*Nitin Saurabh:* N.S. was supported by the seed grant (SG/IITH/F285/2022-23/SG-123) from IIT Hyderabad.

**Acknowledgements** We thank an anonymous reviewer for pointing out that Theorem 5 implies a better quantum communication upper bound for finding a king than the bound given in an earlier version of our paper.

## 1 Introduction

Graph problems have been very widely studied through the lens of query and communication complexity. In the most natural query setting, an algorithm has query access to an oracle that on being input a pair of vertices, outputs whether or not an edge exists between those vertices. In the basic communication complexity setup for graph problems, two parties, say Alice and Bob, are given the information about the edges in  $E_1$  and  $E_2$ , respectively, where  $E_1$  and  $E_2$  are disjoint subsets of all possible edges in the underlying graph. Their task, just as in the query model, is to jointly solve a known graph problem on the graph formed by the edges in  $E_1 \cup E_2$ . Several interesting results are known in these basic query and communication settings in the deterministic, randomized, and quantum models, see, for example, [5, 27, 19, 29, 40, 9, 11] and the references therein.

A prime example of a graph problem whose query complexity and communication complexities have been widely studied is *Graph Connectivity*. The randomized and quantum communication complexities of this problem are known to be  $O(n \log n)$  and  $\Omega(n)$ . This gap has been open for a long time, and the question of closing it has been explicitly asked [29, 27]. On the other hand, its deterministic communication complexity is known to be  $\Theta(n \log n)$  [27].

A graph problem that has been extensively studied in the context of communication complexity is the Clique vs. Independent Set (CIS) problem [47, 25, 26, 8]. The CIS problem is so fundamental that it makes an appearance in the first chapter of standard textbooks on communication complexity [32, 41] (in fact, it is defined on the first page of the latter textbook). The CIS problem is parametrized by a graph  $G = ([n], E)$ , known to both Alice and Bob. Alice is given  $C \subseteq [n]$  that forms a clique in  $G$ , Bob is given  $I \subseteq [n]$  that forms an independent set in  $G$ , and their task is to determine whether or not  $C \cap I = \emptyset$ . Note that if  $C \cap I \neq \emptyset$ , then it must be the case that  $|C \cap I| = 1$ . It was long known that the communication complexity of CIS is  $O(\log^2 n)$  for all graphs  $G$ . More than two decades after this upper bound was discovered, a near-matching lower bound of  $\tilde{\Omega}(\log^2 n)$  was shown to hold for a particular  $G$ , in a culmination of a long line of ground-breaking work [31, 28, 3, 45, 25, 26].

► **Theorem 1** ([47], [26, Theorem 1.2]). *Let  $G$  be an  $n$ -vertex graph. Then,  $D^{\text{cc}}(\text{CIS}_G) = O(\log^2 n)$ . Furthermore, there exists an  $n$ -vertex graph  $G$  such that  $D^{\text{cc}}(\text{CIS}_G) = \tilde{\Omega}(\log^2 n)$ .*

This lower bound on the communication complexity of CIS also gives the currently-best-known lower bound for the exponent in the famous log-rank conjecture [35]. We remark that the upper bound above also holds if the task is to output the label of the unique intersection of  $C$  and  $I$  if  $C \cap I \neq \emptyset$ .

While not as well-studied as the undirected case, communication complexity of directed graph problems has also received some attention in the past (see, for example, [29, 6, 13]). In this work, we consider *tournaments*, which are directed graphs with exactly one directed edge between each pair of vertices (i.e. the underlying undirected graph is complete). We adopt the natural communication complexity setting where Alice knows the orientation of a subset  $E$  of the edges, Bob knows the orientation of the remaining edges, and their goal is to jointly solve a known task on the tournament.

A source of a tournament is a vertex with no in-neighbour. The first problem that we study is source-finding: finding the source of a tournament (if it exists, and reporting that no source exists otherwise). The source-finding problem has recently played a central role in the recent breakthrough by Chattopadhyay, Mande and Sherif that refuted the log approximate-rank conjecture [15] which is the randomized analog of the famous log-rank conjecture [35] of communication complexity. It was also used in the follow-up results [4, 46] that refuted the quantum version of this conjecture. Source-finding has been studied in the context of query complexity and voting theory (see [18] and the references therein). In fact, the problem of finding a source in a tournament (in the bounded-round communication complexity setting) has been studied by Chakrabarti et al. [13, Sections 3, 4] with applications to streaming lower bounds. In a recent preprint, Ghosh and Kuchlous [24] studied the communication complexity of source-finding in general graphs. Interestingly, they showed that source-finding in general directed graphs can be exponentially harder than source-finding in tournaments as demonstrated by our results (Corollary 3).

We denote the source-finding problem in the specific communication setting discussed above by  $\text{SRC}_E$  (recall that  $E$  is the set of edges whose orientation is known to Alice). Perhaps surprisingly, we show that this task is *equivalent* to the CIS problem on undirected graphs.

► **Theorem 2.**

- For all  $n$ -vertex graphs  $G = ([n], E)$ ,  $D^{\text{cc}}(\text{CIS}_G) \leq D^{\text{cc}}(\text{SRC}_E) + O(\log n)$ .
- For all subsets of edges  $E$  of the complete  $n$ -vertex graph, there exists an  $n$ -vertex graph  $G$  such that  $D^{\text{cc}}(\text{SRC}_E) \leq D^{\text{cc}}(\text{CIS}_G)$ .

Using known near-tight bounds on the communication complexity of CIS (Theorem 1), Theorem 2 immediately yields the following corollary which gives near-tight bounds on the communication complexity of finding a source in a tournament.

► **Corollary 3.** For all subsets  $E$  of the edges of the complete  $n$ -vertex graph, the deterministic communication complexity of finding a source of a tournament if it exists, or outputting that there is no source is

$$D^{\text{cc}}(\text{SRC}_E) = O(\log^2 n).$$

Furthermore, there exists a subset  $E$  of edges of the complete  $n$ -vertex graph such that the deterministic communication complexity of finding a source is

$$D^{\text{cc}}(\text{SRC}_E) = \tilde{\Omega}(\log^2 n).$$

We believe that this equivalence between SRC and CIS will generate further insights into relationships among complexity measures in query and communication settings that are yet to be resolved. Recall that the source-finding function was also recently used to refute the randomized and quantum versions of the log-rank conjecture [15, 4, 46]. In particular, these works showed that the randomized and quantum communication complexities of finding a source in a tournament is polynomially large in the input size. However, in their settings, Alice and Bob each know a bit per edge, and that edge’s direction is determined by the bitwise XOR of Alice and Bob’s bits for that edge. In view of this, Corollary 3 demonstrates a fundamental difference between the communication complexities of the source-finding problem when the edge directions are partitioned between Alice and Bob, and when Alice and Bob jointly have partial information about each edge.

Motivated to find a “most-dominant vertex” in a tournament, Landau defined the notion of a *king* in a tournament [34]. A king in a tournament is a vertex  $v$  such that every other vertex  $w$  is either reachable via a path of length 1 or length 2 from  $v$ . While it is easy

to see that there are tournaments that do not have a source, it is also easy to show that every tournament has a king [34, 38]. If a tournament has a source, then it is a unique king in the tournament. In view of this, a natural variant of  $\text{SRC}_E$  (and hence CIS, in view of Theorem 2) is the communication task of finding a king in a tournament.

We remark here that the deterministic query complexity of finding a king in an  $n$ -vertex tournament is still unknown, and the state-of-the-art bounds are  $\Omega(n^{4/3})$  and  $O(n^{3/2})$ , and are from over 2 decades ago [44]. Recently, [37] essentially resolved the randomized and quantum query complexities of this problem: they showed that the randomized query complexity of finding a king in an  $n$ -vertex tournament is  $\tilde{\Theta}(n)$ , and the quantum query complexity is  $\Theta(\sqrt{n})$ . The complexity of finding a king and natural variants of it have also been fairly well-studied in different contexts [44, 2, 10, 33].

We consider the communication complexity of finding a king in an  $n$ -vertex tournament, denoting this task by  $\text{KING}_n$ . Perhaps surprisingly, while resolving the query complexity of finding a king in a tournament seems hard, we are able to essentially resolve its asymptotic deterministic, randomized, and quantum communication complexities.

► **Theorem 4.** *For all disjoint partitions  $E_1, E_2$  of the edges of a tournament, the deterministic, randomized, and quantum communication complexities of finding a king (where Alice knows the edge directions of edges in  $E_1$  and Bob knows the edge directions of edges in  $E_2$ ) are as follows:*

$$\text{D}^{\text{cc}}(\text{KING}_n) = O(n), \quad \text{R}^{\text{cc}}(\text{KING}_n) = O(n), \quad \text{Q}^{\text{cc}}(\text{KING}_n) = O(\sqrt{n} \log n).$$

Furthermore, there exists a disjoint partition  $E_1, E_2$  such that the deterministic, randomized, and quantum communication complexities of finding a king are as follows:

$$\text{D}^{\text{cc}}(\text{KING}_n) = \Omega(n), \quad \text{R}^{\text{cc}}(\text{KING}_n) = \Omega(n), \quad \text{Q}^{\text{cc}}(\text{KING}_n) = \Omega(\sqrt{n}).$$

In order to show our deterministic and randomized upper bounds, we give a  $O(n)$  cost deterministic protocol. Our quantum upper bound follows from the upper bound in Theorem 5 (the upper bound in Theorem 5 is for the problem of finding a vertex of maximum out degree in the same setting, which is always a king [34]). Our lower bounds follow from a carefully constructed reduction from Set-Disjointness. We sketch our proofs in Section 1.1.

Interestingly, our lower bounds actually hold for tournaments that are promised to have exactly 3 kings. It is well known that a tournament cannot have exactly 2 kings [38]. Thus, the only “easier” case than this promised one is that where the input tournament is promised to have exactly one king. This case is handled in Corollary 3 (it is easy to see that a tournament has a unique king iff the unique king is a source in the tournament).

It is folklore [34] that a vertex with maximum out-degree in a tournament is also a king in the tournament. Thus, another natural question that arises is: what is the complexity of finding a maximum out-degree vertex? The deterministic and randomized query complexity of this task is known to be  $\Theta(n^2)$ , and its quantum query complexity is between  $\Omega(n)$  and  $O(n^{3/2})$  [7, 37]. Let  $\text{MOD}_n$  denote the search problem of finding a maximum out-degree vertex in an  $n$ -vertex tournament. We study the communication complexity of  $\text{MOD}_n$ , again in the natural setting where the edges of the tournament are partitioned between Alice and Bob. We show the following:

► **Theorem 5.** *For all disjoint partitions  $E_1, E_2$  of the edges of a tournament, the deterministic, randomized, and quantum communication complexities of finding a maximum out-degree vertex (where Alice knows the edge directions of edges in  $E_1$  and Bob knows the edge directions of edges in  $E_2$ ) are as follows:*

$$\text{D}^{\text{cc}}(\text{MOD}_n) = O(n \log n), \quad \text{R}^{\text{cc}}(\text{MOD}_n) = O(n \log \log n), \quad \text{Q}^{\text{cc}}(\text{MOD}_n) = O(\sqrt{n} \log n).$$

Furthermore, there exist disjoint partitions such that the deterministic, randomized, and quantum communication complexities of finding a maximum out-degree vertex are as follows:<sup>1</sup>

$$D^{\text{cc}}(\text{MOD}_n) = \Omega(n \log n), \quad R^{\text{cc}}(\text{MOD}_n) = \Omega(n), \quad Q^{\text{cc}}(\text{MOD}_n) = \Omega(\sqrt{n}).$$

We direct the reader's attention to the similarity between our communication complexity bounds for  $\text{MOD}_n$  and known bounds for the communication complexity of Graph Connectivity mentioned earlier in this section: just like in that case we are able to give tight bounds on the deterministic communication complexity, but our bounds are loose by logarithmic factors in the randomized and quantum settings.<sup>2</sup> Our randomized and quantum lower bounds follow using exactly the same reduction from Set-Disjointness as in Theorem 4. Our deterministic lower bound follows by a carefully constructed *fooling set* lower bound. We give a sketch of our proofs in Section 1.1.

While most of the relevant literature of finding kings in tournaments deals with minimizing the number of queries to find a king (which is equivalent to minimizing the depth of a decision tree that solves KING), none deal with minimizing the *size complexity* of a decision tree that solves KING. Logarithm of decision tree size complexity is characterized, upto a log factor in the input size, by the *rank* of the underlying relation (see [36] for definition of size), and these are measures that have gained a significant interest in the past few years in various contexts (see, for instance, [14, 17, 16] and the references therein). While the decision tree depth complexity of  $\text{KING}_n$  lies between  $\Omega(n^{4/3})$  and  $O(n^{3/2})$ , we show a tight bound of  $n - 1$  on  $\text{rank}(\text{KING}_n)$ , which implies an  $\Omega(n)$  lower bound and an  $O(n \log n)$  upper bound on the logarithm of decision tree size for  $\text{KING}_n$ . We omit the statement of this result and its proof due to lack of space, and refer the reader to the full version of the paper [36].

## 1.1 Sketch of proofs of main results

### 1.1.1 Equivalence of source-finding and CIS

We first sketch the proof of Theorem 2, which is the equivalence of finding a source in a tournament and the Clique vs. Independent Set problem. Below is a sketch of the proof of the first part of this theorem. Consider a graph  $G = ([n], E)$ , and an input  $C, I$  to the Clique vs. Independent Set problem. Here Bob is given  $C \subseteq [n]$  which is a clique in  $G$ , and Alice is given  $I \subseteq [n]$  which is an independent set in  $G$  (we switch the order of inputs for convenience). Alice and Bob construct the following instance to the source-finding problem:

- Alice has the edge directions of all edges in  $E$ , and Bob has the remaining edge directions in  $\bar{E}$ .
- Alice constructs her edge directions such that all vertices in  $I$  have in-degree 0 with respect to her edge directions in  $E$ . This is easy to do since there are no edges between any pair of vertices in  $I$ . She also ensures that all vertices in  $[n] \setminus I$  have in-degree at least 1, with respect to her edge directions in  $E$ . She can ensure this if  $G$  is a connected graph. (see Section 3.)
- Just as the above, Bob ensures that all vertices in  $C$  have in-degree 0 w.r.t.  $\bar{E}$ , and all vertices in  $[n] \setminus C$  have in-degree at least 1 w.r.t.  $\bar{E}$ .

<sup>1</sup> The edge partition we use to prove our deterministic lower bound is different from the partition we use to prove our randomized and quantum lower bounds.

<sup>2</sup> After a full version of our work appeared in the public domain [36], Ghosh [23] communicated to us a proof of a matching randomized  $\Omega(n \log \log n)$  lower bound in Theorem 5, showing that our randomized upper bound is tight.

Using the properties above, it is not hard to show that  $s = C \cap I$  iff  $s$  is a source in the tournament jointly constructed by Alice and Bob above. This concludes the reduction from CIS to source-finding.

In the other direction, if Alice is given edge directions for the subset  $E$  of edges of the complete  $n$ -vertex graph, then the underlying graph  $G$  that Alice and Bob construct for the CIS problem is  $G = ([n], E)$ . For the purpose of this reduction, we assume that Alice has an independent set as input to CIS, and Bob has a clique. Alice considers her input, an independent set,  $I$  to the CIS problem to be the set of all vertices with in-degree 0 w.r.t.  $E$  (note that these vertices must form an independent set in  $G$ ), and Bob constructs his input clique  $C$  to be all vertices with in-degree 0 w.r.t. his edges (these form a clique w.r.t.  $E$ , and hence in  $G$ ). Note that a source in the initial tournament, if it exists, must be a vertex in  $I \cap C$  since it must have in-degree 0 both w.r.t. Alice's and w.r.t. Bob's edges. Moreover this is the only way in which  $I$  intersection  $C$  is non-empty. In other words,  $I \cap C \neq \emptyset$  iff there is a source in the initial tournament. This concludes the reduction from source-finding to CIS, and hence Theorem 2. Known upper bounds and lower bounds on the communication complexity of the Clique vs. Independent Set problem (Theorem 1) then yield Corollary 3.

Some of our proofs of the lower bounds in Theorems 4 and 5 follow the same outline. In the next section, we sketch our upper bounds, and we sketch our lower bounds in the following section.

### 1.1.2 Upper bounds

We start with ideas behind the upper bounds in Theorem 4. Throughout this paper, we will view a  $n$ -vertex tournament as a string  $G \in \{0, 1\}^{\binom{n}{2}}$ , where the indices are labeled by pairs  $\{i < j \in [n]\}$  and  $G_{i,j} = 1$  means the edge between vertices  $i$  and  $j$  is directed from  $i$  to  $j$ . Recall that the goal is to construct a communication protocol for finding a king in a tournament  $G \in \{0, 1\}^{\binom{n}{2}}$  whose edges are partitioned into  $E_1$  (with Alice) and  $E_2$  (with Bob).

Consider the deterministic communication model. At a high level, our protocol proceeds in rounds, and in each round Alice and Bob reduce the problem to king-finding in a smaller subtournament. In the beginning of each round assume without loss of generality that Alice has a larger number of edges. Alice sends Bob the label of a vertex  $v$  with maximum number of out-neighbours in  $E_1$  along with the in-neighbourhood of  $v$  in  $E_1$  as a bit-string (one bit for every other vertex  $u$  in the current subtournament for which Alice knows the direction of the edge between  $u$  and  $v$ ). Upon receiving  $v$ , Bob also sends the in-neighbourhood of  $v$  in  $E_2$  as a bit-string. Thus both players know the entire in-neighbourhood of  $v$  in the entire tournament by the end of the round. The communication cost so far is at most  $2n + \log n = O(n)$ , where  $n$  is the number of vertices in the current tournament. The players now reduce to finding a king in the in-neighbourhood of  $v$ , since by [38] (also see Lemma 11), this would give a king in the tournament  $G$ . Since  $|E_1| \geq |E_2|$ , the number of out-neighbours of  $v$  is at least  $(n-1)/4$ . This yields a communication protocol of cost  $T(n)$  that is described by a recurrence of the form  $T(n) \leq T(3n/4) + O(n)$ , which is easily seen to give a solution of  $T(n) = O(n)$ . For the quantum upper bound, we note that a maximum out-degree vertex is always a king [34]. Our  $O(\sqrt{n} \log n)$  quantum upper bound for finding a king then immediately follows from Theorem 5, which we describe shortly.

We now sketch proofs of the upper bounds in Theorem 5. Our upper bounds follow from communication protocols for the following problem: Alice and Bob are given  $A \in [n]^n$  and  $B \in [n]^n$ , respectively. Their goal is to output an index  $i \in [n]$  that maximizes  $a_i + b_i$ . We call



this communication problem  $\text{MAXSUM}_{n,n}$ . The reduction from  $\text{MOD}_n$  to  $\text{MAXSUM}_{n,n}$  is easy to see: Alice and Bob construct  $A, B$  to be the vector of out-degrees of all vertices w.r.t. their edges. Thus a deterministic communication protocol of cost  $O(n \log n)$  immediately follows for  $\text{MOD}_n$ : Alice sends  $A$  to Bob, who then computes an answer. We now sketch the randomized upper bound. Let  $S = (s_1, \dots, s_n)$  where  $s_i = a_i + b_i$ . The first observation is that deciding  $s_i \geq s_j$  is equivalent to deciding  $a_i - a_j \geq b_j - b_i$ . The latter can be done with cost  $O(\log \log n)$  and error at most  $1/3$  by using the communication protocol of Greater-Than due to [39, Theorem 1] (see Theorem 21). Thus Alice and Bob have access to a “noisy” oracle that decides whether  $s_i \geq s_j$ , for all  $i, j \in [n]$ , independently with probability at least  $2/3$ . Finding  $\arg \max_{i \in [n]} s_i$  with error probability  $1/3$  can be done by making  $O(n)$  such queries (due to [21], see Theorem 20). This gives a protocol with an overall communication cost of  $O(n \log \log n)$ . The quantum communication protocol is an application of a result of [12], along with a quantum query upper bound for computing  $\arg \max$  (see Theorem 15), see Section 5 for details.

### 1.1.3 Lower bounds

Our intuition for the lower bounds is that a “hard” partition of edges between Alice and Bob should be such that every vertex has an equal number of incident edges with Alice and with Bob. One such natural partition of the edges is as follows: Alice receives the complete tournament restricted to the first  $n/2$  vertices and the complete tournament restricted to the last  $n/2$  vertices, and Bob receives all of the edges between these vertices. While we are unable to use this partition of edges to prove a lower bound for  $\text{KING}_n$ , we do use it to show a deterministic lower bound for  $\text{MOD}_n$ . Our approach to showing a deterministic communication lower bound for  $\text{MOD}_n$  is to construct a large *fooling set* (see Lemma 19). More precisely, for a permutation  $\sigma \in S$ , where  $S$  is a suitably chosen large (size  $2^{\Omega(n \log n)}$ ) subset of  $\mathcal{S}_n$ , we construct inputs  $A_\sigma, B_\sigma$  to Alice and Bob such that vertex 1 is a unique maximum out-degree vertex for all  $\sigma \in S$ . We also ensure that “cross-inputs”  $(A_\sigma, B_{\sigma'})$  with  $\sigma \neq \sigma'$  lead to vertex 1 not being a maximum out-degree vertex as long as  $\sigma$  and  $\sigma'$  are far away in the  $\ell_\infty$  norm, which we force to be true for all permutations in  $S$  by our construction. We refer the reader to Section 5 for technical details.

While we are unable to make the same reduction work to show the communication lower bounds for  $\text{KING}_n$  (and for good reason, since this argument gives an  $\Omega(n \log n)$  lower bound, and there is an  $O(n)$  upper bound for the communication complexity of  $\text{KING}_n$ ) and randomized and quantum communication lower bounds for  $\text{MOD}_n$ , our partition constructed there has a similar flavor to that above. A key intermediate function that we consider for showing our remaining lower bounds is a variant of  $\text{KING}$  inspired by the well-studied Indexing function. Aptly, we name our variant  $\text{IndexKING}$ , defined below. For a tournament  $G \in \{0, 1\}^{\binom{n}{2}}$  with vertex set  $[n]$ , and a set  $S \subseteq [n]$ , we use the notation  $G|_S$  to denote the subtournament of  $G$  induced on the vertices in  $S$ .

► **Definition 6.** Let  $n > 0$  be a positive integer. Define the  $\text{IndexKING}_n$  communication problem as follows: Alice is given a set  $S \subseteq [n]$  and Bob is given a tournament  $G \in \{0, 1\}^{\binom{n}{2}}$  on  $n$  vertices. Their goal is to output a king in  $G|_S$ .

We consider the restriction of  $\text{IndexKING}$  to those inputs where Bob’s tournament is a transitive tournament (see Definition 12). We denote this variant by  $\text{t-IndexKING}$ . A moment’s observation (see Observation 8) reveals that this problem is equivalently formulated as follows. We name this version the *Permutation Maximum Finding* problem, defined below, and we believe that this problem is of independent interest.

► **Definition 7** (Permutation Maximum Finding). *Let  $n > 0$  be a positive integer. In the Permutation Maximum Finding problem,  $\text{PMF}_n$ , Alice is given as input a subset  $S$  of  $[n]$ , Bob is given a permutation  $\sigma \in \mathcal{S}_n$ , and their goal is to output*

$$\text{PMF}_n(S, \sigma) = \begin{cases} \perp & S = \emptyset \\ \arg \max_{j \in S} \sigma(j) & S \neq \emptyset. \end{cases}$$

Unless explicitly mentioned otherwise, we assume that Alice’s input  $S$  to  $\text{PMF}_n$  is always a non-empty set. In other words, in the PMF problem, Alice is given a subset of  $[n]$ , Bob is given a ranking of all elements in  $[n]$  (here,  $\sigma(i)$  denotes the rank of  $i$ ), and their goal is to find the element in Alice’s set that has the largest rank.

► **Observation 8.** *Let  $n > 0$  be a positive integer. Then,  $\text{cost}(\text{PMF}_n) = \text{cost}(\text{t-IndexKING}_n)$ , where  $\text{cost} \in \{\text{D}^{\text{cc}}, \text{R}^{\text{cc}}, \text{Q}^{\text{cc}}\}$ .<sup>3</sup>*

We refer the reader to the full version [36] for a proof.

We show that Set-Disjointness reduces to PMF (see Lemma 28 and its proof). The lower bound results for PMF follow from known results for communication complexity of Set-Disjointness (see Theorem 17).

Next we reduce from  $\text{PMF}_n$  to KING. Our reduction ensures that an instance  $(S, \sigma)$  to  $\text{PMF}_n$  gives us a tournament  $G_{S, \sigma}$  with the following properties:

- The tournament has  $3n$  vertices, partitioned into  $V_1, V_2, V_3$ , of  $n$  vertices each, each labeled by elements of  $[n]$ . The internal edges (edges in  $\binom{V_1}{2}, \binom{V_2}{2}$  and  $\binom{V_3}{2}$ ) in each of the partitions are with Bob, and these correspond to transitive tournaments defined by  $\sigma$ .
- The remaining “cross” edges are all with Alice, and the directions of these are determined by  $S$  (see Figure 1 for details).
- The tournament  $G_{S, \sigma}$  has exactly three kings (which are also the three unique maximum out-degree vertices), one in each  $V_i$ , and each of these is labeled by  $\text{PMF}_n(S, \sigma)$ .

Thus finding a king or a maximum out-degree vertex in  $G_{S, \sigma}$  amounts to Alice and Bob solving  $\text{PMF}_n$ , which we’ve already sketched to be hard via a reduction from Set-Disjointness. An interesting point to note is that this actually shows a lower bound on the communication complexity of finding a king, even when the input tournament is promised to have exactly three kings. Recall that we showed that finding a king can be done with  $O(\log^2 n)$  deterministic communication when an input is promised to have exactly one king (Corollary 3). Also it is easy to show using Lemma 11 that there are no tournaments with exactly two kings. Thus, the “easiest” non-trivial case of a promised tournament with exactly three kings is already hard for communication.

## 2 Preliminaries

Let  $[n] = \{1, \dots, n\}$ . We use the notation  $\text{polylog}(n)$  to denote  $O(\log(n)^c)$  for some fixed constant  $c$ . For  $f : \mathbb{N} \rightarrow \mathbb{N}$ , we use the notation  $\tilde{O}(f)$  to denote  $O(f \log^{c_1} f)$  and  $\tilde{\Omega}(f)$  to denote  $\Omega(f / (\log^{c_2} f))$ , for some constants  $c_1, c_2$ .

A tournament  $G \in \{0, 1\}^{\binom{[n]}{2}}$  is a complete directed graph on  $n$ -vertices. For  $v, w \in [n]$  such that  $v < w$ , if  $G_{v, w} = 1$  then there is an out-edge from  $v$  to  $w$ , i.e.  $v \rightarrow w$  (otherwise there is an out-edge from  $w$  to  $v$ ). In this case we say that  $v$  1-step dominates  $w$ . Similarly, for  $u, w \in [n]$ , if there exists a  $v \in [n]$  such that  $u \rightarrow v$  and  $v \rightarrow w$  then we say that  $u$  2-step

<sup>3</sup> We actually prove the stronger statement that the problems  $\text{PMF}_n$  and  $\text{t-IndexKING}_n$  are equivalent, in the sense that Alice and Bob need not communicate to go one from one problem to another.

dominates  $w$ . Let  $S \subseteq [n]$  be such that  $v$  2-step (1-step) dominates  $w$  for all  $w \in S$ . We then say that  $v$  2-step (1-step) dominates  $S$ . It is easy to see that there are tournaments where no vertex 1-step dominates all other vertices (such a vertex is called the *source* of  $G$ ). However, it is now folklore that every tournament has a vertex  $v$  such that every vertex  $w \neq v$  is either 1-step or 2-step dominated by  $v$ . Such a vertex is called a *king* of the tournament (see [34]).

► **Lemma 9** (Folklore). *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a tournament. Then there exists a vertex  $v \in [n]$  such that  $v$  is a king of  $G$ .*

For a vertex  $v \in [n]$ , let  $N^-(v) = \{w \in [n] : w \rightarrow v\}$  and  $N^+(v) = \{w \in [n] : v \rightarrow w\}$ . Thus  $N^-(v)$  and  $N^+(v)$  denote the in-neighbourhood and out-neighbourhood of  $v$  in  $G$ , respectively. The in-degree of  $v$ , denoted by  $d^-(v)$  is defined as  $|N^-(v)|$ , and similarly the out-degree of  $v$  is denoted by  $d^+(v)$  and is defined as  $|N^+(v)|$ . If a vertex has maximum out-degree in the tournament, then that vertex is a king of the tournament (a proof can be found in [38]).

► **Lemma 10** ([34]). *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a tournament and  $v \in [n]$  be a vertex of maximum out-degree in  $G$ . Then  $v$  is a king in  $G$ .*

For  $S \subseteq [n]$  let  $G|_S$  be the tournament induced on  $S$  by  $G$ , i.e.  $G|_S$  is a tournament with vertex set as  $S$  and direction of edges in  $S$  are same as that in  $G$ .

The following is an important lemma that we use often.

► **Lemma 11** ([38]). *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a tournament and  $v \in [n]$ . If a vertex  $u$  is a king in  $G|_{N^-(v)}$ , then  $u$  is a king in  $G$ .*

A special class of tournaments is the class of transitive tournaments, which we define next.

► **Definition 12** (Transitive Tournament). *A tournament  $G \in \{0, 1\}^{\binom{n}{2}}$  is transitive if it satisfies the following property: for all  $u, v, w \in [n]$ ,  $u \rightarrow v$  and  $v \rightarrow w$  implies  $u \rightarrow w$ .*

In other words, a transitive tournament is a tournament which is a directed acyclic graph.

► **Lemma 13** (Properties of Transitive Tournaments). *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a transitive tournament. There is an ordering  $v_1, \dots, v_n$  of  $[n]$  such that*

- $v_1$  is a source vertex and hence a unique king in  $G$ , and
- for all  $i \in \{2, \dots, n\}$ ,  $v_i$  is source vertex in  $G|_{[n] \setminus \bigcup_{j=1}^{i-1} \{v_j\}}$ .

**Proof.** Since  $G$  is a directed acyclic graph, a topological sort on the vertices gives a source of the graph. Let this vertex be  $v_1$ . The vertex  $v_i$  is obtained by applying the same argument over the transitive tournament  $G|_{[n] \setminus \bigcup_{j=1}^{i-1} \{v_j\}}$ . ◀

## 2.1 Query and Communication Complexity

We refer the reader to the full version [36] of our paper for the formal setup of deterministic, randomized, and quantum query complexity.

► **Definition 14** ( $\text{ARGMAX}_{k,n}$ ). *Let  $k$  be a positive integer and let  $a \in ([k])^n$ . Given query access to  $a$ , find  $i \in [n]$  such that  $a_i \geq a_j$  for all  $j \neq i \in [n]$ .*

► **Theorem 15** ([20]). *There exists a quantum query algorithm for  $\text{ARGMAX}_{k,n}$  with query cost  $O(\sqrt{n})$ .*

## 64:10 On the Communication Complexity of Finding a King in a Tournament

We refer the reader to the full version [36] of our paper for the formal setup of deterministic, randomized, and quantum communication complexity.

► **Definition 16** (Set-Disjointness). *Let  $n > 0$  be a positive integer. The Set-Disjointness problem is denoted by  $\text{DISJ}_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$  and is defined by*

$$\text{DISJ}_n(A, B) = 1 \iff A \cap B = \emptyset,$$

where  $A, B \subseteq [n]$  are the characteristic sets of Alice and Bob's inputs, respectively.

The communication complexity of  $\text{DISJ}_n$  is extensively studied. We require the following known bounds on its communication complexity [5, 30, 43, 42, 1].

► **Theorem 17** (Communication complexity of Set-Disjointness). *The deterministic, randomized, and quantum communication complexity of  $\text{DISJ}_n$  is as follows:*

$$D^{\text{cc}}(\text{DISJ}_n) = n, \quad R^{\text{cc}}(\text{DISJ}_n) = \Theta(n), \quad Q^{\text{cc}}(\text{DISJ}_n) = \Theta(\sqrt{n}).$$

It is a folklore result that, classically, query algorithms for functions give communication protocols for these functions composed with small gadgets with very little blowup in the complexity. In the quantum setup we have the following theorem, that essentially follows from [12].

► **Theorem 18** ([12]). *Let  $f \subseteq \mathcal{D}_f^n \times \mathcal{R}$  be a relation where  $\mathcal{D}_f = [k]$  for some finite  $k$ , and let  $g : \mathcal{D}_g \times \mathcal{D}_g \rightarrow \mathcal{D}_f$  be a function. For all  $\varepsilon > 0$ , if  $Q_\varepsilon(f) \leq T$  then  $Q_\varepsilon^{\text{cc}}(f \circ g) \leq 2T(\lceil \log n \rceil + \lceil \log k \rceil + \lceil \log |\mathcal{D}_g| \rceil)$ .*

We refer the reader to the full version [36] of our paper for a proof.

A *fooling set* for a communication problem  $f \subseteq (\mathcal{X} \times \mathcal{Y}) \times \mathcal{R}$  is a set  $S \subseteq \mathcal{X} \times \mathcal{Y}$  such that for all pairs  $s_1 = (x_1, y_1)$  and  $s_2 = (x_2, y_2)$  in  $S$ , we have

$$\{r \in \mathcal{R} \mid (x_1, y_1, r) \in f \wedge (x_1, y_2, r) \in f \wedge (x_2, y_1, r) \in f \wedge (x_2, y_2, r) \in f\} = \emptyset.$$

► **Lemma 19.** *Let  $f \subseteq (\mathcal{X} \times \mathcal{Y}) \times \mathcal{R}$  be a communication problem, and let  $S \subseteq \mathcal{X} \times \mathcal{Y}$  be a fooling set for  $f$ . Then,  $D^{\text{cc}}(f) \geq \log |S|$ .*

We refer the reader to standard texts for a formal proof [32, Lemma 1.20]. We remark that standard texts usually frame the fooling set lower bound as a lower bound technique for communication complexity of functions rather than relations, but the same proof technique is easily seen to show the statement above as well. A sketch of the proof is as follows: The leaves of a protocol tree of depth  $c$  yields a partition of the space  $\mathcal{X} \times \mathcal{Y}$  into  $2^c$  rectangles, each of which has at least one  $r \in \mathcal{R}$  that is a valid output for all pairs of inputs in the rectangle. By the property of a fooling set, each element of it must belong to a different leaf. This implies the number of leaves in any protocol for  $f$  must be at least  $|S|$ , implying that the depth of any protocol must be at least  $\log |S|$ .

We require the following theorem that gives an algorithm to find the maximum in a list given noisy comparison oracle access. The formulation we use below follows easily from [21, Theorem 15].

► **Theorem 20** ([21, Theorem 15]). *Let  $S = (s_1, \dots, s_n)$  be a list of  $n$  numbers. Suppose we have access to a “noisy” oracle, that takes as input a pair of indices  $i \neq j \in [n]$ , and outputs a bit that equals  $\mathbb{I}[s_i \geq s_j]$  with probability at least  $2/3$ , independent of the outputs to the other queries. Then there is an algorithm that makes  $O(n)$  queries to the noisy oracle and outputs  $\arg \max_{i \in [n]} s_i$  with probability at least  $2/3$ .*

► **Theorem 21** ([39, Theorem 1]). *Let  $n > 0$  be a positive integer. The  $\text{GT} : [n] \times [n] \rightarrow \{0, 1\}$ , where Alice is given  $x \in [n]$  and Bob is given  $y \in [n]$  is defined as  $\text{GT}(x, y) = 1$  if and only if  $x \geq y$ . The randomized communication complexity of  $\text{GT}$  is  $O(\log \log n)$ .*

## 2.2 Formal definitions of graph problems of interest

For clarity and completeness, we include formal definitions of the tasks of finding a king and finding a maximum out-degree vertex in this section.

► **Definition 22.** *Let  $n > 0$  be a positive integer. Define  $\text{KING}_n \subseteq \{0, 1\}^{\binom{n}{2}} \times [n]$  to be*

$$(G, v) \in \text{KING}_n \iff v \text{ is a king in the tournament } G.$$

► **Definition 23.** *Let  $n > 0$  be a positive integer. Define  $\text{MOD}_n \subseteq \{0, 1\}^{\binom{n}{2}} \times [n]$  to be*

$$(G, v) \in \text{MOD}_n \iff v \text{ is a maximum out-degree vertex in the tournament } G.$$

When we give communication upper bounds for these problems, our upper bounds hold for all partitions of the input variables  $\binom{n}{2}$  between Alice and Bob. When we give lower bounds, we exhibit specific partitions for which our lower bounds hold.

## 3 Communication complexity of finding a source

We consider the communication complexity of finding a source in a tournament if it exists. Alice knows the edge directions of a subset  $E_A$  of the edges of a tournament  $T \in \{0, 1\}^{\binom{n}{2}}$ , Bob knows the directions of the remaining edges  $E_B$ , and their goal is to output the label of a source in the whole tournament if it exists, or output that the tournament has no source. Formally, for a partition of edges  $E_A, E_B$  of the complete  $n$ -vertex graph, define

$$\text{SRC}_{E_A} : \{0, 1\}^{E_A} \times \{0, 1\}^{E_B} \rightarrow \{0, 1, \dots, n\} \quad (1)$$

to be  $\text{SRC}_{E_A}(a, b) = 0$  if there is no source in the tournament defined by edge directions  $a, b$ , and  $\text{SRC}_{E_A}(a, b) = i$  if vertex  $i$  is the (unique) source in the same tournament. We define the decision version of this problem to be  $\text{SRC}_{E_A}^{\text{dec}} : \{0, 1\}^{E_A} \times \{0, 1\}^{E_B} \rightarrow \{0, 1\}$ . That is,  $\text{SRC}_{E_A}^{\text{dec}}$  outputs 0 if there is no source in the tournament, and outputs 1 if there is a source.

Below, we define the celebrated Clique vs. Independent Set problem on an  $n$ -vertex graph  $G$  [47], which we henceforth abbreviate as  $\text{CIS}_G$ . The  $\text{CIS}_G$  problem is associated with an  $n$ -vertex undirected graph  $G = (V, E)$ . In this problem, Alice and Bob both know  $G$ . Alice is given as input a clique  $x \subseteq [n]$  in  $G$ , Bob is given as input an independent set  $y \subseteq [n]$ , and their goal is to either output that  $x \cap y = \emptyset$ , or output the label of the (unique) vertex  $v$  with  $\{v\} = x \cap y$ .<sup>4</sup>

There has been a plethora of work on the Clique vs. Independent set problem, see for example, [47, 25, 26, 8]. Of relevance to us is Theorem 1, which gives near-tight bounds on the deterministic communication complexity of this problem.

Perhaps surprisingly, we show that the communication problem of finding a source in a tournament is *equivalent* to the Clique vs. Independent Set problem. Corollary 3 would then immediately follow. We now prove Theorem 2.

<sup>4</sup> Conventionally, the Clique vs. Independent Set problem is phrased as a decision problem, where the task is to determine if  $x \cap y$  is empty or non-empty. The known bounds we state here are easily seen to hold for the “search version” that we consider as well.

## 64:12 On the Communication Complexity of Finding a King in a Tournament

**Proof of Theorem 2.** In this proof, we assume for convenience that in the Clique vs. Independent Set Problem, Alice is given an independent set and Bob is given a clique.

- Let  $G = (V, E)$  be an  $n$ -vertex graph. Let  $I, C \subseteq [n]$  be Alice and Bob's input to  $\text{CIS}_G$ , respectively. Recall that the vertices in  $I$  form an independent set in  $G$  and the vertices in  $C$  form a clique in  $G$ . We now describe the reduction from  $\text{CIS}_G$  to  $\text{SRC}_E$ . Before delving into the main reduction, we do a preprocessing of small communication cost to make sure that  $G$  is connected and the size of the independent set  $I$  is at least 3.

Preprocessing: Bob sends the label of the connected component in  $G$  that his clique  $C$  is part of. Alice removes from her independent set  $I$ , all vertices that aren't part of this connected component. She now sends a bit to Bob to indicate whether  $|I| \geq 3$ . If not, she further sends labels of the two vertices in  $I$  to Bob who then responds with an answer. This requires a total of  $O(\log n)$  communication cost. We can therefore assume that the graph  $G$  is connected and  $|I| \geq 3$  for the rest of the reduction. Alice and Bob locally construct the following inputs to  $\text{SRC}_E$  (recall that Alice must construct edge directions in  $E$ , and Bob must construct the remaining edge directions).

- Alice orients the edges in  $E$ , using Claim 24 and the fact that  $G$  is a connected graph, such that only the vertices in  $I$  have in-degree 0.
- Bob orients the edges in  $\bar{E}$  as follows. For vertices in  $C$ , he orients the edges in their connected components in  $\bar{G}$ , using Claim 24, such that only the vertices in  $C$  have in-degree 0. Next he orients the edges of connected components that don't contain vertices of  $C$ . If this connected component is not a tree, he uses Claim 25 to orient the edges such that no vertex has in-degree 0. If the connected component is a tree, he orients the edges in an arbitrary way.

Let  $T$  denote the tournament constructed above. We next show that  $(I, C)$  is a 1-input to  $\text{CIS}_G$  iff there exists a source in  $T$ . This would prove the first part of the theorem. Moreover, we show that when there is a source in the constructed tournament, the source vertex is the same as the unique vertex in  $I \cap C$ .

Let  $(I, C)$  be a 1-input to  $\text{CIS}_G$  and  $s$  be the unique vertex in  $I \cap C$ . We show that  $s$  is the source in the tournament  $T$ . By construction, the neighbours of  $s$  in  $E$  are the outneighbours of  $s$  in Alice's input, and the neighbours of  $s$  in  $\bar{E}$  are the outneighbours of  $s$  in Bob's input.

We prove the contrapositive for the other direction. Let  $(I, C)$  be a 0-input to  $\text{CIS}_G$ , i.e.,  $I \cap C = \emptyset$ . We show that there is no source in  $T$ . Vertices in  $\bar{I}$  are ruled out from being a source by the orientation of Alice's edges. Now the vertices of  $I$  form a clique in Bob's input, thus they form a connected component that is not a tree (since  $|I| \geq 3$ ). Since this connected component does not contain a single vertex from  $C$  (since we assumed  $I \cap C = \emptyset$ ), the construction above (using Claim 25) implies that all vertices in  $I$  have in-degree at least 1 w.r.t. Bob's edge directions. Thus, there is no source in the entire tournament.

- In the other direction, let  $\{0, 1\}^{E_A}$  and  $\{0, 1\}^{E_B}$  be Alice and Bob's input to  $\text{SRC}_{E_A}$ , where  $E_A, E_B$  form a partition of the edges of the  $n$ -vertex complete graph. Say that the tournament formed by these inputs is  $T$ . Alice and Bob construct the following instance to the Clique vs. Independent Set problem.
  - The graph is  $G = (V, E)$  with  $V = [n]$  and  $E = E_A$ .
  - Alice constructs  $I \subseteq [n]$  to be all of the vertices with in-degree 0 w.r.t.  $E_A$ . It is easy to see that  $I$  forms an independent set in  $G$  since any edge between vertices in  $I$  causes one of the vertices in  $I$  to have in-degree at least 1.
  - Bob constructs  $C \subseteq [n]$  to be all of the vertices with in-degree 0 w.r.t.  $E_B$ . As in the previous bullet, it is easy to see that  $C$  forms an independent set in  $\bar{G}$ , and hence a clique in  $G$ .

Consider the input  $(I, C)$  to  $\text{CIS}_G$  as constructed above. We show now that  $I \cap C \neq \emptyset$  iff there is a source in  $T$ , which would prove the second part of the theorem since  $(I, C)$  and  $G$  were constructed using no communication.

Suppose  $s$  is a source in  $T$ . Since  $s$  has in-degree 0 w.r.t. both  $E_A$  and  $E_B$ , we must have  $s \in I \cap C$ . Moreover, since every other vertex must have in-degree at least 1, such a vertex is either not in  $I$  or not in  $C$ . Thus,  $s = I \cap C$ . In the other direction, suppose  $s = I \cap C$ . By the construction above,  $s$  must have in-degree 0 w.r.t. both  $E_A$  and  $E_B$ , and hence is a source in  $T$ . ◀

▷ **Claim 24.** Let  $T$  be a tree,  $V$  be its vertex set and  $I$  be an independent set in  $T$ . Then there exists an orientation of the edges of  $T$  such that exactly the vertices in  $V \setminus I$  have in-degree at least 1.

*Proof of Claim 24.* We now show a procedure to orient the edges such that the set of vertices with in-degree 0 equals the set  $I$ . Consider a (left-to-right) listing of subsets of vertices based on their distances from the set  $I$ . So if the listing looks like  $V_0, V_1, \dots, V_j, \dots$ , then  $V_0 = I$ , and  $V_j \subseteq V \setminus I$  is the set of vertices such that the length of a shortest path to reach a vertex in  $I$  equals  $j$ . We orient the edges from  $V_i \rightarrow V_{i+1}$  for  $i \geq 0$ . The edges within a partition, say  $V_i$ , are oriented arbitrarily. Now using the fact that tree is a connected graph, it is easily seen that every vertex in  $V \setminus I$  has in-degree at least 1. Moreover, by our construction, all vertices in  $V_0 = I$  has in-degree 0. ◀

▷ **Claim 25.** Let  $G$  be a connected graph that is not a tree. Then, there exists an orientation of the edges of  $G$  such that every vertex of  $G$  has in-degree at least 1.

*Proof of Claim 25.* Since  $G$  is connected but not a tree, it contains a cycle, say  $C$ . Orient the edges of  $C$  in a cyclic way to give in-degree 1 to every vertex in  $C$ , and then orient the edges “away” from the cycle  $C$  (in a manner similar to the proof in Claim 24 where  $V_0 = C$  here) to add 1 to in-degrees of vertices in  $V \setminus C$ . Thus the directed graph so constructed has no vertex with in-degree 0. ◀

## 4 Communication complexity of KING

The proof of Theorem 4 is divided into two parts. We show the upper bounds in Section 4.1 and the lower bounds in Section 4.2.

### 4.1 Upper bounds on communication complexity of $\text{KING}_n$

We start by proving an  $O(n)$  upper bound on the deterministic communication complexity which also implies an  $O(n)$  upper bound on the randomized communication complexity.

► **Lemma 26.** *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a tournament and let  $E_1, E_2$  be a partition of the edges of  $G$ . The deterministic and randomized communication complexity of finding a king of  $G$ , where Alice is given  $E_1$  and Bob is given  $E_2$ , is upper bounded as follows*

$$D^{\text{cc}}(\text{KING}_n) = O(n), \quad R^{\text{cc}}(\text{KING}_n) = O(n).$$

## 64:14 On the Communication Complexity of Finding a King in a Tournament

**Proof.** The proof follows via the Protocol in Algorithm 1.

■ **Algorithm 1** Deterministic Communication Protocol for  $\text{KING}_n$ .

---

```

1: Input: Let  $G \in \{0,1\}^{\binom{n}{2}}$  be a tournament and  $E_1, E_2 \subseteq \{(i,j) : i < j \in [n]\}$  be a
   partition of the edges of  $G$ . Alice (Player 1) is given  $\{0,1\}^{E_1}$  and Bob (Player 2) is given
    $\{0,1\}^{E_2}$ .
2:  $S = [n]$ 
3: while  $|E_1| > n$  and  $|E_2| > n$  do
4:    $b \leftarrow \arg \max_{i \in \{0,1\}} |E_i|$  ▷ Ties broken arbitrarily
5:    $v \leftarrow \arg \max_{w \in [n]} \{\text{out-degree}(w) \text{ in } E_b\}$  ▷ Ties broken arbitrarily
6:   Player  $b$  sends to Player  $1 - b$  the label of  $v$  along with a  $|S|$ -bit indicator vector of
   the in-neighbourhood of  $v$  in  $E_b$ 
7:   Player  $1 - b$  sends an  $|S|$ -bit indicator vector of the in-neighbourhood of  $v$  in  $E_{1-b}$ 
8:    $S \leftarrow S \cap N^-(v)$ 
9:    $E_1 \leftarrow$  the edges of  $E_1$  that are present in  $G|_S$ 
10:   $E_2 \leftarrow$  the edges of  $E_2$  that are present in  $G|_S$ 
11: end while
12: if  $|E_1| \leq n$  then
13:   Alice sends  $E_1$  to Bob
14:   Bob outputs a king of the tournament.
15: else if  $|E_2| \leq n$  then
16:   Bob sends  $E_1$  to Alice
17:   Alice outputs a king of the tournament.
18: end if

```

---

**Correctness.** It is easy to see that in every iteration of the **while** loop, the size of either  $E_1$  or  $E_2$  decreases by at least 1. This shows that our algorithm always terminates.

Let  $S^{(i)}$  denote the set  $S$  in  $i$ 'th iteration of the **while** loop, where  $S^{(1)} = [n]$ . We maintain the invariant that in every iteration of the **while** loop, a king in  $G|_{S^{(i+1)}}$  is also a king in  $G|_{S^{(i)}}$ . This follows easily from Lemma 11 since  $S^{(i+1)}$  is obtained from  $S^{(i)}$  by restricting to vertices in the in-neighbourhood of some vertex  $v$  in Line 8. Assume without loss of generality that the **while** loop terminates with  $|E_1| \leq n$ . In this case, in Line 13, Alice sends her edges to Bob who outputs a king of  $G$ .

**Cost.** We show that the cost of Protocol 1 is upper bounded by  $O(n)$  for all tournaments  $G \in \{0,1\}^{\binom{n}{2}}$ . Suppose we enter the **while** loop with  $|S| = k$ . Let  $c(k)$  be the number of bits communicated during the execution of the **while** loop. Consider Line 6, and assume without loss of generality that  $|E_1| \geq |E_2|$ , thus  $|E_1| \geq (1/2 \cdot \binom{k}{2})$ . Since every edge in  $E_1$  is an out-edge for some vertex (note that  $E_1$  and  $E_2$  are subsets of edges of  $G|_S$  due to Line 9 and Line 10) we have  $\sum_{u \in S} d^+(u) \geq (1/2 \cdot \binom{k}{2})$  (where the out-degrees are only computed in  $E_1$ ) and hence by an averaging argument there exists  $v \in S$  such that the out-degree of  $v$  when restricted to  $E_1$  (and therefore  $S$ ) is at least  $(k-1)/4$ . Thus the in-degree of  $v$  in  $S$  is at most  $(3/4 \cdot (k-1))$ . Furthermore, in each iteration of the **while** loop,  $\lceil \log k \rceil + k$  bits are communicated in Line 6 and  $k$  bits are communicated in Line 7. We have the following upper bound on  $c(k)$ :  $c(k) \leq c(3k/4) + \lceil \log k \rceil + 2k$ , and thus  $c(n) = O(n)$ . Also observe that either Line 13 or Line 16 is executed and in each case at most  $n$  bits are communicated. Thus the overall number of bits communicated is  $O(n)$ . ◀



Next, we give an  $O(\sqrt{n} \log n)$  cost quantum communication protocol for  $\text{KING}_n$ . Our quantum communication upper bound is a corollary of Theorem 5 which gives a quantum communication protocol for finding a maximum out-degree vertex in a tournament (such a vertex is also a king, see Lemma 10).

► **Lemma 27.** *Let  $G \in \{0, 1\}^{\binom{n}{2}}$  be a tournament and let  $E_1, E_2$  be a partition of  $E$ . The quantum communication complexity, where Alice is given  $E_1$  and Bob is given  $E_2$ . Then*

$$Q^{\text{cc}}(\text{KING}_n) = O(\sqrt{n} \log n).$$

## 4.2 Lower bounds on communication complexity of $\text{KING}_n$

Next, we prove the lower bound. In order to do this, we first give a lower bound on the communication complexity of  $\text{PMF}_n$ . Recall that, in this problem, Alice is given as input a subset  $S$  of  $[n]$ , Bob is given a ranking of elements of  $[n]$  defined by  $\sigma$ , and their goal is to output the element in  $S$  that has the largest rank according to  $\sigma$ .

► **Lemma 28.** *The deterministic, randomized, and quantum communication complexity of  $\text{PMF}_n$  is as follows:*

$$D^{\text{cc}}(\text{PMF}_n) = \Omega(n), \quad R^{\text{cc}}(\text{PMF}_n) = \Omega(n), \quad Q^{\text{cc}}(\text{PMF}_n) = \Omega(\sqrt{n}).$$

**Proof.** We show that Set-Disjointness reduces to  $\text{PMF}_n$  and the lemma follows from Theorem 17. We describe the reduction next.

Consider an input to Set-Disjointness,  $S, T \subseteq [n]$  where  $S$  is with Alice and  $T$  is with Bob. Alice and Bob locally construct the following instance of  $\text{PMF}_n$ : Alice retains her set  $S$ , and Bob creates an arbitrary  $\sigma$  such that the following holds:

$$\forall i \neq j \in [n], \quad (T_i = 0) \wedge (T_j = 1) \implies \sigma(i) < \sigma(j).$$

In other words, Bob creates a permutation  $\sigma$  of  $[n]$  that ranks all of the indices in  $T$  higher than all of the indices outside  $T$ . They then run a protocol for  $\text{PMF}_n$  with inputs  $S, \sigma$ , let  $k$  be the output of this protocol. If  $k \in T$  then they return  $S \cap T \neq \emptyset$  else they return  $S \cap T = \emptyset$ .

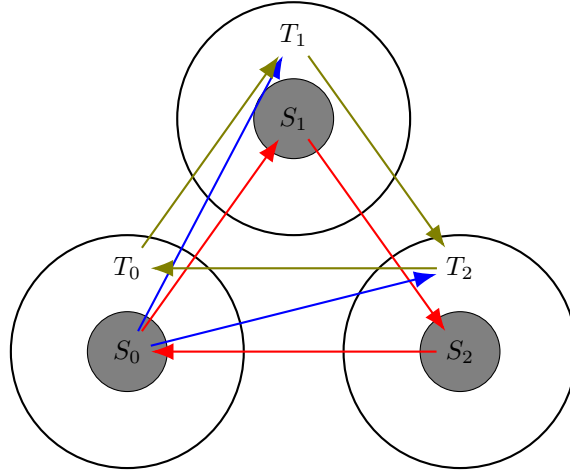
**Correctness.** If  $\text{PMF}_n(S, \sigma) = \perp$ , then the players know (without any additional communication) that  $S = \emptyset$  and hence  $\text{DISJ}_n(S, T) = 1$ . Thus, we may assume  $S \neq \emptyset$ . Since any protocol for  $\text{PMF}_n$  must output an index  $k \in S$ . By Bob's construction of  $\sigma$ , the elements of  $T$  are ranked higher than elements that are not in  $T$ . Since  $k$  is the output of a protocol for  $\text{PMF}_n$ ,  $k$  is the highest ranked element in  $S$  by  $\sigma$ . Thus if  $k$  is not among the top  $|T|$  ranked elements, then all elements of  $S$  are ranked lower than all elements of  $T$  (by Bob's construction of  $\sigma$ ) and  $S \cap T = \emptyset$ . On the other hand if  $k$  is among the top  $|T|$  ranked elements then  $k \in T \cap S$ . These conditions can be checked by Bob who has  $\sigma$  and  $k$ . ◀

By the equivalence of  $\text{PMF}$  and the transitive variant of  $\text{IndexKING}$  (Observation 8), Lemma 28 implies the same lower bounds on  $\text{t-IndexKING}_n$ .

We thus immediately conclude the same lower bounds on the general  $\text{IndexKING}$  problem (where Bob's tournament is arbitrary, and need not be transitive).

► **Corollary 29.** *The deterministic, randomized, and quantum communication complexity of  $\text{IndexKING}_n$  is as follows:*

$$D^{\text{cc}}(\text{IndexKING}_n) = \Omega(n), \quad R^{\text{cc}}(\text{IndexKING}_n) = \Omega(n), \quad Q^{\text{cc}}(\text{IndexKING}_n) = \Omega(\sqrt{n}).$$



- **Figure 1** Visual depiction of  $G_{S, \sigma}$ . For each  $b \in \{0, 1, 2\}$ ,  $S_b$  contains the vertices  $\{i_b : i \in S\}$  and  $T_b$  contains the vertices  $\{i_b : i \notin S\}$ . There are four types of edges (also see Definition 30):
- Edges of **Type 1** are those within each  $T_b \cup S_b$ , here  $i_b \rightarrow j_b$  iff  $\sigma(i) > \sigma(j)$ .
  - Edges of **Type 2** are those between  $S_b$  and  $T_{b'}$  for  $b \neq b'$ , here  $i_b \rightarrow j_{b'}$ .
  - Edges of **Type 3** are those between  $S_b$  and  $S_{b'}$  for  $b \neq b'$ , here  $i_b \rightarrow j_{b'}$  iff  $b' = b + 1 \pmod{3}$ .
  - Edges of **Type 4** are those between  $T_b$  and  $T_{b'}$  for  $b \neq b'$ , here  $i_b \rightarrow j_{b'}$  iff  $b' = b + 1 \pmod{3}$ .

We now give a lower bound on the communication complexity of  $\text{KING}_n$ . For this we first define a class of tournaments that we use in our proof.

### 4.3 A class of tournaments

In this section, we define a special class of tournaments on  $3n$  vertices, that are parametrized by a subset  $S \subseteq [n]$  and an ordering  $\sigma$  of  $[n]$ .

► **Definition 30.** Given a set  $S \subseteq [n]$  and  $\sigma \in \mathcal{S}_n$ , define the tournament  $G_{S, \sigma}$  on  $3n$  vertices as follows:

- The vertex set is  $V = \{i_b : i \in [n], b \in \{0, 1, 2\}\}$ .
- For each  $b \in \{0, 1, 2\}$  and all  $i \neq j \in [n]$ , the direction of the edge between  $i_b$  and  $j_b$  is  $i_b \rightarrow j_b$  iff  $\sigma(i) > \sigma(j)$ . We refer to these as **Type 1** edges.
- For all  $b \neq b' \in \{0, 1, 2\}$ , all  $i \in S$  and all  $j \notin S$ ,  $i_b \rightarrow j_{b'}$  is an edge. We refer to these as **Type 2** edges.
- For all  $b \neq b' \in \{0, 1, 2\}$  and all  $i \neq j \in S$ , the direction between the edge  $i_b$  and  $j_{b'}$  is  $i_b \rightarrow j_{b'}$  iff  $b' = b + 1 \pmod{3}$ . We refer to these as **Type 3** edges.
- For all  $b \neq b' \in \{0, 1, 2\}$  and all  $i \neq j \notin S$ , the direction between the edge  $i_b$  and  $j_{b'}$  is  $i_b \rightarrow j_{b'}$  iff  $b' = b + 1 \pmod{3}$ . We refer to these as **Type 4** edges.

We refer the reader to Figure 1 for a pictorial representation and some additional notation.

► **Lemma 31.** Let  $n > 0$  be a positive integer,  $S \subseteq [n]$  and  $\sigma \in \mathcal{S}_n$ . Then, the tournament  $G_{S, \sigma}$  has exactly three kings, namely  $k_0, k_1, k_2$ , where  $k = \arg \max_{j \in S} \sigma(j)$ . Moreover,  $k_0, k_1, k_2$  are the only vertices with maximum out-degree in  $G_{S, \sigma}$ .

**Proof.** We first show that  $k_0$  is a king. The argument for  $k_1, k_2$  being kings follows similarly. To show that  $k_0$  is a king, we exhibit paths of length one or two from  $k_0$  to all other vertices in the tournament.

- First note that for any element  $j \in S$ , there is an edge from  $k_0$  to  $j_0$  since  $k = \arg \max_{j \in S} \sigma(j)$  (this is an edge of Type 1). Thus,  $k_0$  1-step dominates  $S_0$ .
- For all  $j \notin S$  and  $b \in \{1, 2\}$ , there is an edge (of Type 2) from  $k_0$  to  $j_b$ . Thus,  $k_0$  1-step dominates  $T_1$  and  $T_2$ .
- For  $j, j' \in S$ , there is an edge (of Type 3) from  $k_0$  to  $j_1$ . Thus  $k_0$  1-step dominates  $S_1$ . There is also an edge (also of Type 3) from  $j_1$  to  $j'_2$ . Thus,  $k_0$  2-step dominates  $S_2$ .
- For an arbitrary  $j \in S$ , as noted above, there is an edge from  $k_0$  to  $j_1$ . For  $j' \notin S$ , there is an edge (of Type 2) from  $j_1$  to  $j'_0$ . Thus,  $k_0$  2-step dominates  $T_0$ .

This shows that  $k_0$  (and similarly  $k_1$  and  $k_2$ ) is a king in  $G_{S,\sigma}$ .<sup>5</sup> We next show that no other vertex is a king. We do this by showing for every other vertex  $k'_b$ , a vertex that is not 1-step or 2-step dominated by  $k'_b$ .

- Consider  $k' \neq k \in S$  and  $b \in \{0, 1, 2\}$ . We now show that  $k'_b$  does not 1-step or 2-step dominate  $k_b$ .
  - Since  $k_b$  is the unique king in the transitive tournament  $(G_{S,\sigma})|_{S_b}$  (see Lemma 13),  $k'_b$  does not 1-step dominate  $k_b$  via Type 1 edges. Moreover, the only vertices that are 1-step dominated by  $k'_b$  via Type 1 edges are a subset of vertices in  $S_b \cup T_b$ . None of these vertices can 1-step dominate  $k_b$  since  $(G_{S,\sigma})|_{S_b \cup T_b}$  is a transitive tournament. This shows that  $k'_b$  cannot 1-step dominate or 2-step dominate  $k_b$  by first using an edge of Type 1.
  - The only other out-going edges from  $k'_b$  are either of Type 2 or Type 3.
  - Consider a Type 2 edge which goes from  $k'_b$  to  $T_{b+1 \pmod 3}$  ( $T_{b+2 \pmod 3}$  follows similarly). By construction, there is no edge from any vertex in  $T_{b+1 \pmod 3}$  to  $k_b$  (see Figure 1).
  - Now consider a Type 3 edge which goes from  $k'_b$  to  $S_{b+1 \pmod 3}$ . By construction, there is no edge from any vertex in  $S_{b+1 \pmod 3}$  to  $k_b$  (see Figure 1).
- Consider  $k' \notin S$  and  $b \in \{0, 1, 2\}$ . We now show that  $k'_b$  does not 1-step or 2-step dominate  $k_{b+2 \pmod 3}$ .
  - The only out-going edges from  $k'_b$  are either of Type 1 or Type 4. On taking a Type 1 edge,  $k'_b$  can only 1-step dominate a subset of vertices of  $S_b \cup T_b$ . None of these vertices have an edge to  $k_{b+2 \pmod 3}$  (see Figure 1). Thus,  $k'_b$  cannot 2-step dominate  $k_{b+2 \pmod 3}$  by first taking a Type 1 edge.
  - A Type 4 edge goes from  $k'_b$  to a vertex in  $T_{b+1 \pmod 3}$ . By construction, no vertex in  $T_{b+1 \pmod 3}$  has an edge to  $k_{b+2 \pmod 3}$  (see Figure 1).

Finally, we observe that  $k_0, k_1, k_2$  are the only three vertices with maximum out-degree in  $G_{S,\sigma}$ . Observe that the out-degrees of  $k_0, k_1, k_2$  are all equal by symmetry. By Lemma 10, a vertex with maximum out-degree in  $G_{S,\sigma}$  is a king in  $G_{S,\sigma}$ . This, along with the proof above that shows that  $k_0, k_1, k_2$  are the only kings in  $G_{S,\sigma}$ , immediately implies that  $k_0, k_1, k_2$  are the only three vertices with maximum out-degree in  $G_{S,\sigma}$ . ◀

#### 4.4 Proof of Theorem 4

We now prove Theorem 4. The upper bounds follow from the arguments in Section 4.1. For the lower bounds, we do a reduction from PMF. The class of tournaments constructed in Section 4.3, and its properties, play a crucial role in the reduction.

---

<sup>5</sup> We remark here that there is an alternative proof that shows  $k_0$  to be a king: consider an arbitrary  $j_1$  for an arbitrary  $j \in S$ . The in-neighborhood of  $j_1$  contains  $S_0$  and a subset of  $S_1 \cup T_1$ . It can be verified that  $k_0$  is a source (and hence a king) in the tournament restricted to the in-neighbourhood of  $j_1$ . Lemma 11 then implies that  $k_0$  is a king. We choose to keep the current proof for clarity.

**Proof of Theorem 4.** The upper bounds follow from Lemma 26 and Lemma 27.

For the lower bounds, consider an input  $S \subseteq [n]$  to Alice and  $\sigma \in \mathcal{S}_n$  to Bob for  $\text{PMF}_n$ . Alice and Bob jointly construct the tournament  $G_{S,\sigma}$ . Note that this construction is completely local and involves no communication; Alice can construct all edges of Types 2, 3 and 4, and Bob can construct all edges of Type 1 (see Figure 1). By Lemma 31, there are exactly 3 kings in  $G_{S,\sigma}$  and these are  $\{i_b : b \in \{0, 1, 2\}, i = \arg \max_{j \in S} \sigma(j) = \text{PMF}_n(S, \sigma)\}$  (recall Definition 7). Thus, running a protocol for  $\text{KING}_{3n}$  on input  $G_{S,\sigma}$  (where Alice has edges of Types 2, 3 and 4, and Bob has edges of Type 1) gives the solution to  $\text{PMF}_n(S, \sigma)$  at no additional cost. Lemma 28 implies the required lower bounds. ◀

## 5 Communication complexity of MOD

Recall that in the  $\text{MOD}_n$  communication problem, Alice and Bob are given inputs in  $\{0, 1\}^{E_1}$  and  $\{0, 1\}^{E_2}$ , respectively, where  $E_1$  and  $E_2$  form a partition of the edge set  $\binom{[n]}{2}$ . Their goal is to output a vertex  $v$  that has maximum out-degree in the tournament formed by the union of their edges. We next prove Theorem 5. In this theorem we settle the communication complexity of finding a maximum out-degree vertex in a tournament in the deterministic, randomized, and quantum models, up to logarithmic factors in the input size. In the deterministic model we are able to show a tight  $\Theta(n \log n)$  bound.

We first define an intermediate communication problem,  $\text{MAXSUM}_{n,k}$ , which seems independently interesting to study from the perspective of communication complexity.

► **Definition 32.** Let  $n, k > 0$  be positive integers. In the  $\text{MAXSUM}_{n,k}$  problem, Alice is given  $A = (a_1, \dots, a_n) \in [k]^n$ , Bob is given  $B = (b_1, \dots, b_n) \in [k]^n$ , and their goal is to output  $\arg \max_{j \in [n]} (a_j + b_j)$  (if there is a tie, they can output any of the tied indices).

$\text{MAXSUM}_{n,k}$  is easily seen to be the composition of two problems: the outer problem is  $\text{ARGMAX}_{2k,n}$  (see Definition 14) and the inner function is  $\text{SUM}_k$  (which adds two integers in  $[k]$ , one with Alice and the other with Bob). It is also easy to see that  $\text{MOD}_n$  reduces to  $\text{MAXSUM}_{n,2n}$ : Alice and Bob can locally construct  $(a_1, \dots, a_n)$  and  $(b_1, \dots, b_n)$  to be the out-degree vectors of all the vertices restricted to edges in their inputs. Thus, a cost- $c$  protocol for  $\text{MAXSUM}_{n,2n}$  also gives a protocol for  $\text{MOD}_n$ .

We note here that our upper bounds (Theorem 5) actually give upper bounds for the more general  $\text{MAXSUM}_{n,k}$  problem; the deterministic, randomized, and quantum communication upper bounds here are  $O(n \log k)$ ,  $O(n \log \log k)$  and  $O(\sqrt{n} \log k \log n)$ , respectively. Next, we proceed to give a proof of Theorem 5.

**Proof of Theorem 5.** For the upper bounds, we exhibit protocols of the required cost for  $\text{MAXSUM}_{n,n}$ , which is only a (potentially) harder problem.

- For the deterministic upper bound, note that Alice can just send her input to Bob with cost  $n \log n$ , and Bob can output the answer.
- The randomized upper bound follows by using Theorem 20 with the list  $s = (a_1 + b_1, \dots, a_n + b_n)$ , and observing that testing whether  $a_i + b_i \geq a_j + b_j$  can be done with communication  $O(\log \log n)$  and success probability at least  $2/3$  (Theorem 21).
- For the quantum upper bound, recall that  $\text{MAXSUM}_{n,n}$  is the composition of  $\text{ARGMAX}_{2n,n}$  (with an input list in  $[2n]^n$ ) and  $\text{SUM}$  (sum of 2 integers in  $[n]$ , one with Alice and the other with Bob). Here,  $\text{ARGMAX}_{2n,n}$  has query complexity  $O(\sqrt{n})$ , where query access is to the values of the elements of the list (see Theorem 15) and  $\text{SUM} : [n] \times [n] \rightarrow [2n]$ . Setting  $\mathcal{D}_g = [n]$ ,  $\mathcal{D}_f = [2n]$ ,  $g = \text{SUM}_n : \mathcal{D}_g \times \mathcal{D}_g \rightarrow \mathcal{D}_f$ ,  $f = \text{ARGMAX}_{2n,n} \subseteq \mathcal{D}_f^n \times [n]$  in Theorem 18, this gives a quantum communication upper bound of  $O(\sqrt{n} \log n)$ .

**Randomized and quantum lower bounds.** The randomized and quantum lower bounds follow the same proof as that of Theorem 4 (see Section 4.4) because the three kings in  $G_{S,\sigma}$  are precisely the maximum out-degree vertices there as well (see Lemma 31). This argument also shows a deterministic lower bound of  $\Omega(n)$ .

**Deterministic lower bound.** We now turn our attention to the deterministic lower bound of  $\Omega(n \log n)$ , which does not use the same reduction as in the proof of Theorem 4. We show this via a fooling set argument (Lemma 19). Below, we assume that the first half of Alice’s input corresponds to the out-degree sequence of a tournament on vertex set  $L = \{1, 2, \dots, n/2\}$ , the second half of her input corresponds to the out-degree sequence of a tournament on vertex set  $R = \{1', 2', \dots, (n/2)'\}$ , and Bob’s input is the out-degree sequence of the complete bipartite tournament between  $L$  and  $R$ . We focus on inputs that are induced by tournaments of the following form, that are defined for a permutation  $\sigma \in \mathcal{S}_{n/2-1}$  that acts in an identical fashion on  $\{2, 3, \dots, n/2\}$  and  $\{2', 3', \dots, (n/2)'\}$ . We call Alice and Bob’s input constructed below  $A_\sigma$  and  $B_\sigma$ , respectively.

1. Vertex 1 is the source in  $L$ , and vertex  $1'$  is the source in  $R$ . These edges are with Alice.<sup>6</sup>
2. Vertex 1 has edges towards  $1'$  and  $\sigma^{-1}(2')$ . All other vertices in  $\{3', 4', \dots, (n/2)'\}$  have edges pointing towards vertex 1. These edges are with Bob.
3. For all  $i, j \in \{2, 3, \dots, n/2\}$ , there is an edge from  $i$  to  $j$  iff  $\sigma(i) < \sigma(j)$ . Similarly there is an edge from  $i'$  to  $j'$  iff  $\sigma(i') < \sigma(j')$ . These edges are with Alice.
4. For  $i \in \{2, 3, \dots, n/2\}$ , there is an edge from  $i$  to  $1'$ . These edges are with Bob.
5. For  $i, j \in \{2, 3, \dots, n/2\}$ , there is an edge from  $i$  to  $j'$  iff  $\sigma(i) \leq \sigma(j)$ . These edges are with Bob.

We now verify that vertex 1 is the unique vertex with maximum out-degree in the whole tournament (and hence the first coordinate must be output in the corresponding inputs to Alice and Bob for  $\text{MOD}_n$ ).

- Items 1 and 2 above ensure that vertex 1 has out-degree  $n/2 - 1 + 2 = n/2 + 1$ .
- Item 1 and Item 4 ensure that the out-degree of vertex  $1'$  is  $n/2 - 1$ .
- Item 1 and Item 5 ensure that vertex  $\sigma^{-1}(2')$  has out-degree  $n/2 - 2$ .
- For  $i \in \{2, 3, \dots, n/2\}$ , the out-degree of vertex  $\sigma^{-1}(i)$  is  $n/2 - i$  from Alice’s input (Item 3) plus  $i$  from Bob’s input (Item 5), which gives a total of  $n/2$ .
- For  $i \in \{3, 4, \dots, n/2\}$ , the out-degree of vertex  $\sigma^{-1}(i')$  is  $n/2 - i$  from Alice’s input (Item 3) plus  $i - 1$  from Bob’s input (Item 5), which gives a total of  $n/2 - 1$ .

These bullets verify that for input  $(A_\sigma, B_\sigma)$ , vertex 1 is the unique maximum out-degree vertex. Our fooling set will be of the form  $F = \{(A_\sigma, B_\sigma) : \sigma \in S\}$ , where  $S \subseteq \mathcal{S}_{n/2-1}$  is chosen appropriately. The property that  $S$  will satisfy is that for all  $\sigma \neq \sigma' \in S$ , at least one of the inputs  $(A_\sigma, B_{\sigma'})$  or  $(A_{\sigma'}, B_\sigma)$  will *not* have vertex 1 as a maximum out-degree vertex. We will also construct  $S$  such that  $|S| = 2^{\Omega(n \log n)}$ . Lemma 19 will then imply the required deterministic communication lower bound of  $\Omega(n \log n)$ .

It remains to construct  $S \subseteq \mathcal{S}_{n/2-1}$ , which we do in the remaining part of this proof. We construct  $S$  such that it satisfies the following property.

$$\forall \sigma \neq \sigma' \in S, \quad \exists i \in \{2, 3, \dots, n/2\} : |\sigma(i) - \sigma'(i)| \geq 2.$$

<sup>6</sup> When we say “edges are with Alice/Bob”, we actually mean Alice/Bob’s out-degree of vertices is determined by the directions of the underlying edges. In this case we mean Alice’s first coordinate is  $n/2 + 1$  because vertex 1 is a source in  $L$ .

In the two bullets below, we first show why such an  $S$  satisfies the required fooling set property, and then show a construction of  $S$  of size  $2^{\Omega(n \log n)}$ .

- Let  $\sigma \neq \sigma'$  be an arbitrary pair of elements of  $S$ . Without loss of generality, assume that  $i \in \{2, 3, \dots, n/2\}$  is such that  $\sigma'(i) - \sigma(i) \geq 2$  (otherwise switch the roles of  $\sigma$  and  $\sigma'$  and run the same argument). Consider the input  $(A_\sigma, B_{\sigma'})$ . Note that the out-degree of vertex 1 remains  $n/2 + 1$  because all edges incident on it are fixed for all inputs in our fooling set. Alice's contribution to the out-degree of vertex  $i$  is  $n/2 - \sigma(i)$ , and Bob's contribution is  $\sigma'(i)$ , which gives a total of  $n/2 + \sigma'(i) - \sigma(i) \geq n/2 + 2$ . Thus vertex 1 cannot be a maximum out-degree vertex in the input  $(A_\sigma, B_{\sigma'})$ .
- We construct such an  $S$  greedily one element at a time. At any step in the construction we maintain the invariant that the current set  $T$  satisfies

$$\forall \sigma \neq \sigma' \in T, \quad \exists i \in \{2, 3, \dots, n/2\} : |\sigma(i) - \sigma'(i)| \geq 2.$$

Additionally we maintain a “candidate” set of permutations in  $\mathcal{S}_{n/2-1}$  that are not in  $T$ , and have the property that adding any of them to  $T$  will satisfy  $T$ 's invariant. Initially we start with  $T = \emptyset$  and the candidate set as  $\mathcal{S}_{n/2-1}$ , which clearly satisfies the required invariant. At any stage, after adding  $\sigma$  to  $T$ , we remove the set  $S_\sigma$  from the candidate set, where  $S_\sigma$  is defined as

$$S_\sigma := \{\tau \in \mathcal{S}_{n/2-1} : |\tau(i) - \sigma(i)| < 2 \ \forall i \in \{2, 3, \dots, n/2\}\}.$$

It is easy to verify by induction that  $T$  and the candidate set thus constructed always satisfy the required invariant. The initial size of the candidate set is  $(n/2-1)! = 2^{\Omega(n \log n)}$ , and at each step we are removing at most  $3^n$  elements from the candidate set. This means that the number of iterations of this construction is at least  $2^{\Omega(n \log n - n)} = 2^{\Omega(n \log n)}$ , which is what we needed. ◀

We remark that while it may seem like the argument used in the previous proof may be adaptable to prove a deterministic communication lower bound of  $\Omega(n \log n)$  for  $\text{KING}_n$ , this is not possible in view of our  $O(n)$  deterministic communication upper bound for  $\text{KING}_n$  from Theorem 4. This shows an inherent difference between  $\text{MOD}_n$  and  $\text{KING}_n$  in the setting of deterministic communication complexity.

► **Remark 33.** We note that our  $\Omega(n \log n)$  lower bound for  $\text{MOD}_n$  also solves Problem 2 in [22].

---

## References

- 1 Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory Comput.*, 1(1):47–79, 2005. doi:10.4086/TQC.2005.V001A004.
- 2 Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. *ACM Trans. Algorithms*, 12(2):19:1–19:19, 2016. doi:10.1145/2701427.
- 3 Kazuyuki Amano. Some improved bounds on communication complexity via new decomposition of cliques. *Discrete Applied Mathematics*, 166:249–254, 2014. doi:10.1016/j.dam.2013.09.015.
- 4 Anurag Anshu, Naresh Goud Boddur, and Dave Touchette. Quantum log-approximate-rank conjecture is also false. In *Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 982–994, 2019. doi:10.1109/FOCS.2019.00063.
- 5 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 337–347, 1986. doi:10.1109/SFCS.1986.15.

- 6 Yakov Babichenko, Shahar Dobzinski, and Noam Nisan. The communication complexity of local search. In *Symposium on Theory of Computing, (STOC)*, pages 650–661, 2019. doi:10.1145/3313276.3316354.
- 7 R. Balasubramanian, Venkatesh Raman, and G. Srinivasaragavan. Finding scores in tournaments. *J. Algorithms*, 24(2):380–394, 1997. doi:10.1006/JAGM.1997.0865.
- 8 Kaspars Balodis, Shalev Ben-David, Mika Göös, Siddhartha Jain, and Robin Kothari. Unambiguous DNFs and Alon-Saks-Seymour. In *Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 116–124. IEEE, 2021. doi:10.1109/FOCS52979.2021.00020.
- 9 Gal Beniamini and Noam Nisan. Bipartite perfect matching as a real polynomial. In *Symposium on Theory of Computing, (STOC)*, pages 1118–1131. ACM, 2021. doi:10.1145/3406325.3451002.
- 10 Arindam Biswas, Varunkumar Jayapaul, Venkatesh Raman, and Srinivasa Rao Satti. Finding kings in tournaments. *Discret. Appl. Math.*, 322:240–252, 2022. doi:10.1016/j.dam.2022.08.014.
- 11 Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 1174–1185, 2022. doi:10.1109/FOCS54457.2022.00113.
- 12 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Symposium on the Theory of Computing, (STOC)*, pages 63–68, 1998. doi:10.1145/276698.276713.
- 13 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Symposium on Discrete Algorithms, (SODA)*, pages 1786–1802, 2020. doi:10.1137/1.9781611975994.109.
- 14 Arkadev Chattopadhyay, Yogesh Dahiya, Nikhil S. Mande, Jaikumar Radhakrishnan, and Swagato Sanyal. Randomized versus deterministic decision tree size. In *Symposium on Theory of Computing, (STOC)*, pages 867–880, 2023. doi:10.1145/3564246.3585199.
- 15 Arkadev Chattopadhyay, Nikhil S. Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. *J. ACM*, 67(4):23:1–23:28, 2020. doi:10.1145/3396695.
- 16 Arjan Cornelissen, Nikhil S. Mande, and Subhasree Patro. Improved quantum query upper bounds based on classical decision trees. In *Foundations of Software Technology and Theoretical Computer Science, (FSTTCS)*, volume 250, pages 15:1–15:22, 2022. doi:10.4230/LIPICS.FSTTCS.2022.15.
- 17 Yogesh Dahiya and Meena Mahajan. On (simple) decision tree rank. *Theor. Comput. Sci.*, 978:114177, 2023. doi:10.1016/J.TCS.2023.114177.
- 18 Palash Dey. Query complexity of tournament solutions. In *Conference on Artificial Intelligence, (AAAI)*, pages 2992–2998, 2017. doi:10.1609/AAAI.V31I1.10702.
- 19 Pavol Duris and Pavel Pudlák. On the communication complexity of planarity. In *Fundamentals of Computation Theory, (FCT)*, volume 380 of *Lecture Notes in Computer Science*, pages 145–147, 1989. doi:10.1007/3-540-51498-8\_14.
- 20 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *CoRR*, quant-ph/9607014, 1996. URL: <http://arxiv.org/abs/quant-ph/9607014>.
- 21 Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Computing with unreliable information (preliminary version). In *Symposium on Theory of Computing, (STOC)*, pages 128–137. ACM, 1990. doi:10.1145/100216.100230.
- 22 Maxime Flin and Parth Mittal.  $(\Delta+1)$  vertex coloring in  $O(n)$  communication. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*, pages 416–424. ACM, 2024. doi:10.1145/3662158.3662796.
- 23 Prantar Ghosh. Private Communication, 2024.
- 24 Prantar Ghosh and Sahil Kuchlous. New algorithms and lower bounds for streaming tournaments. *CoRR*, abs/2405.05952, 2024. doi:10.48550/arxiv.2405.05952.

- 25 Mika Göös. Lower bounds for clique vs. independent set. In *Symposium on Foundations of Computer Science, (FOCS)*, pages 1066–1076, 2015. doi:10.1109/FOCS.2015.69.
- 26 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM J. Comput.*, 47(6):2435–2450, 2018. doi:10.1137/16M1059369.
- 27 András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *Symposium on Theory of Computing, (STOC)*, pages 186–191, 1988. doi:10.1145/62212.62228.
- 28 Hao Huang and Benny Sudakov. A counterexample to the Alon-Saks-Seymour conjecture and related problems. *Comb.*, 32(2):205–219, 2012. doi:10.1007/S00493-012-2746-4.
- 29 Gábor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. New bounds on the classical and quantum communication complexity of some graph properties. In *Foundations of Software Technology and Theoretical Computer Science, (FSTTCS)*, volume 18, pages 148–159, 2012. doi:10.4230/LIPICS.FSTTCS.2012.148.
- 30 Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992. doi:10.1137/0405044.
- 31 Eyal Kushilevitz, Nathan Linial, and Rafail Ostrovsky. The Linear-Array Conjecture in Communication Complexity is False. *Comb.*, 19(2):241–254, 1999. doi:10.1007/S004930050054.
- 32 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.
- 33 Oded Lachish, Felix Reidl, and Chhaya Trehan. When you come at the king you best not miss. In *Foundations of Software Technology and Theoretical Computer Science, (FSTTCS)*, volume 250, pages 25:1–25:12, 2022. doi:10.4230/LIPICS.FSTTCS.2022.25.
- 34 H.G. Landau. On dominance relations and the structure of animal societies: III The condition for a score structure. *The bulletin of mathematical biophysics*, 15:143–148, 1953. doi:10.1007/BF02476378.
- 35 László Lovász and Michael E. Saks. Lattices, mobius functions and communications complexity. In *Symposium on Foundations of Computer Science, FOCS*, pages 81–90, 1988. doi:10.1109/SFCS.1988.21924.
- 36 Nikhil S Mande, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the communication complexity of finding a king in a tournament. *CoRR*, 2024. arXiv:2402.14751.
- 37 Nikhil S. Mande, Manaswi Paraashar, and Nitin Saurabh. Randomized and quantum query complexities of finding a king in a tournament. In *Foundations of Software Technology and Theoretical Computer Science, (FSTTCS)*, volume 284, pages 30:1–30:19, 2023. doi:10.4230/LIPICS.FSTTCS.2023.30.
- 38 Stephen B Maurer. The king chicken theorems. *Mathematics Magazine*, 53(2):67–80, 1980.
- 39 Noam Nisan. The communication complexity of threshold gates. *Combinatorics, Paul Erdos is Eighty*, 1:301–315, 1993.
- 40 Noam Nisan. The demand query model for bipartite matching. In *Symposium on Discrete Algorithms, (SODA)*, pages 592–599, 2021. doi:10.1137/1.9781611976465.36.
- 41 Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.
- 42 Alexander Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya of the Russian Academy of Sciences, mathematics*, 67(1):159–176, 2003.
- 43 Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992. doi:10.1016/0304-3975(92)90260-M.
- 44 Jian Shen, Li Sheng, and Jie Wu. Searching for sorted sequences of kings in tournaments. *SIAM J. Comput.*, 32(5):1201–1209, 2003. doi:10.1137/S0097539702410053.
- 45 Manami Shigeta and Kazuyuki Amano. Ordered biclique partitions and communication complexity problems. *Discrete Applied Mathematics*, 184:248–252, 2015. doi:10.1016/j.dam.2014.10.029.



- 46 Makrand Sinha and Ronald de Wolf. Exponential separation between quantum communication and logarithm of approximate rank. In *Annual Symposium on Foundations of Computer Science, (FOCS)*, pages 966–981, 2019. doi:10.1109/FOCS.2019.00062.
- 47 Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. Syst. Sci.*, 43(3):441–466, 1991. doi:10.1016/0022-0000(91)90024-Y.



# Capacity-Achieving Gray Codes

Venkatesan Guruswami     
University of California, Berkeley, CA, USA

Hsin-Po Wang     
University of California, Berkeley, CA, USA

---

## Abstract

To ensure differential privacy, one can reveal an integer fuzzily in two ways: (a) add some Laplace noise to the integer, or (b) encode the integer as a binary string and add iid BSC noise. The former is simple and natural while the latter is flexible and affordable, especially when one wants to reveal a sparse vector of integers. In this paper, we propose an implementation of (b) that achieves the capacity of the BSC with positive error exponents. Our implementation adds error-correcting functionality to Gray codes by mimicking how software updates back up the files that are getting updated (“coded Gray code”). In contrast, the old implementation of (b) interpolates between codewords of a black-box error-correcting code (“Grayed code”).

**2012 ACM Subject Classification** Mathematics of computing → Coding theory; Security and privacy → Privacy-preserving protocols

**Keywords and phrases** Gray codes, capacity-achieving codes, differential privacy

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.65

**Category** RANDOM

**Funding** *Venkatesan Guruswami*: NSF grant CCF-2210823 and a Simons Investigator Award

## 1 Introduction

Differential privacy is the art of publishing collective facts without leaking any detail of any user. A mathematically rigorous way to do so is adding noise to an aggregation function that is Lipschitz continuous (sometimes of bounded variation) in every argument. More concretely, suppose that we are interested in a feature  $\varphi: \{0, 1\}^n \rightarrow [m]$  that satisfies

$$|\varphi(u) - \varphi(u')| \leq 1, \text{ for } u := (u_1, \dots, u_i, \dots, u_n) \text{ and } u' := (u_1, \dots, 1 - u_i, \dots, u_n),$$

i.e., changing the data of the  $i$ th user does not change the feature too much. Then publishing  $\varphi(u) + L$ , where  $L$  follows the Laplace distribution with decay rate  $\varepsilon$ , is  $\varepsilon$ -differentially private [3]. That is,

$$\text{Prob}\{\varphi(u) + L < t\} \leq \exp(\varepsilon) \text{Prob}\{\varphi(u') + L < t\} \quad (1)$$

for any number  $t \in \mathbb{R}$ , meaning that a data broker will have a hard time telling if  $u_i$  is 0 or 1.

Publishing  $\varphi(u) + L$  is called the *Laplace mechanism* [3]. It is optimal<sup>1</sup> privacy-wise as (1) assumes equality half of the time. But it turns out to be randomness-costly and space-inefficient when we have many features  $\varphi_1, \dots, \varphi_\ell$  to publish, wherein only  $k \ll \ell$  of them are non-zero<sup>2</sup> for a given  $x$ . In this case, the Laplace mechanism will add noise to

---

<sup>1</sup> Note that we can always choose to publish  $\varphi(u) + 100L$ , which is more private than  $\varphi(u) + L$  by being less informative and less useful. We say that  $\varphi(u) + L$  is optimal because it strikes a balance between (1) and utility.

<sup>2</sup> For example,  $\varphi_i(u)$  could be the number of times the  $i$ th English word was mentioned in a forum archive  $u$ . Most word counts are going to be zero.





|              |              |
|--------------|--------------|
| $g^1 = 0000$ | $\rho_1 = 1$ |
| $g^2 = 1000$ | $\rho_2 = 2$ |
| $g^3 = 1100$ | $\rho_3 = 1$ |
| $g^4 = 0100$ | $\rho_4 = 3$ |
| $g^5 = 0110$ | $\rho_5 = 1$ |
| $g^6 = 1110$ | $\rho_6 = 2$ |
| $g^7 = 1010$ | $\rho_7 = 1$ |
| $g^8 = 0010$ | $\rho_8 = 4$ |
| $g^9 = 0011$ |              |

are the first nine strings. (Digits that are flipped are highlighted.)

A *robust Gray code* [1, 6] encodes integers as binary strings such that they can be fuzzily recovered even if some bits are erased or corrupted. Given the motivational Figure 1, let us use the binary symmetric channels (BSC) with crossover probability  $p \in (0, 1/2)$  to model the errors. Then a robust Gray code is a pair of encoder

$$\mathcal{E}: [m] \rightarrow \{0, 1\}^{1 \times n}$$

and decoder

$$\mathcal{D}: \{0, 1\}^{1 \times n} \rightarrow [m]$$

such that (a)  $\mathcal{E}(x)$  and  $\mathcal{E}(x+1)$  differ by one bit and (b)

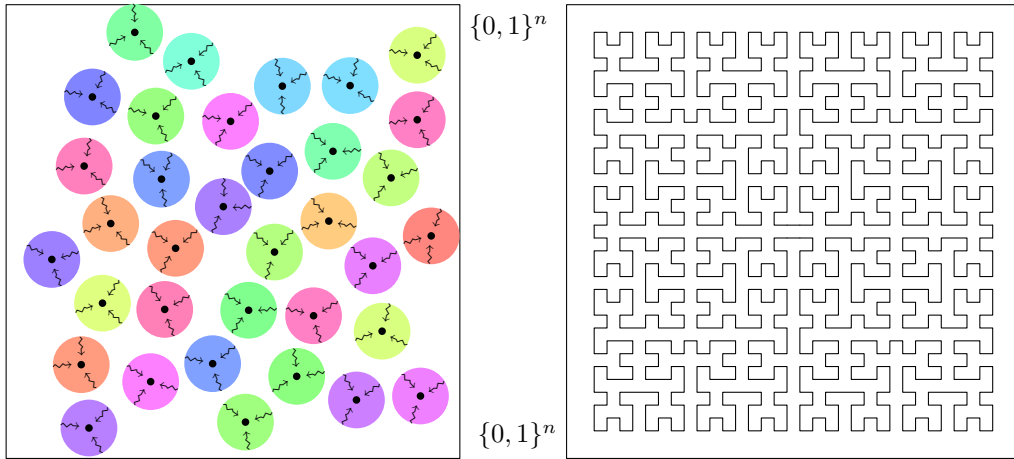
$$\text{Prob}\left\{\left|\mathcal{D}(\text{BSC}_p^n(\mathcal{E}(x))) - x\right| > t\right\} < 2^{-\Omega(n)} + 2^{-\Omega(t)} \quad (3)$$

for all  $x \in [m-1]$  and all  $t > 1$ . Here,  $\text{BSC}_p^n$  flips each of the  $n$  bits with probability  $p$ . Note that (3) is almost as good as the Laplace mechanism in that  $2^{-\Omega(t)}$  decays exponentially in  $t$ . The only catch is that when  $t \gg n$ , the other error term  $2^{-\Omega(n)}$  dominates  $2^{-\Omega(t)}$ . This  $2^{-\Omega(n)}$  is unavoidable because there is always<sup>3</sup> a  $2^{-\mathcal{O}(n)}$  chance that BSC will flip all ones to zero.

Apart from robustness, we also care about space efficiency. We know that, by Shannon's theory, the code rate  $\log_2(m)/n$  cannot exceed the capacity of  $\text{BSC}_p$ , which is  $1 + p \log_2(p) - (1-p) \log_2(1-p)$ . But how close can they be? Before our work, Lolck and Pagh's construction [6] achieves 1/4 of the capacity (1/3 in [6, Appendix A]) and Fathollahi and Wootters's construction [4] achieves 1/2 of the capacity. This means that the latter uses half of the space to achieve the same privacy level.

In this work, and in a concurrent work by Con, Fathollahi, Gabrys, and Yaakobi [2], we will show that the capacity can be achieved. This means that, subject to the framework of Figure 1, the tradeoff between privacy and space is now asymptotically tight. We also show that our code has linear encoding and decoding complexity, meaning that even the speed cannot be significantly improved.

<sup>3</sup> Note that we implicitly assume that  $p$  is bounded away from zero. This is a common practice in coding theory where channel parameters,  $p$  in this case, are fixed while the other parameters vary. Also note that  $\text{BSC}_p$  here plays the role of the Laplace noise, so it would make less sense to have  $p$  too close to zero unless, of course, one is aiming for some special privacy regime.



(a) An error-correcting code is some points that can be decoded up to some radius.

(b) A gray code is a Hamiltonian cycle that only goes in the cardinal directions.

■ **Figure 2** Figurative illustrations of error-correcting codes and Gray codes.

## 1.2 Previous approaches

Earlier works [6, 4] baked robust Gray codes with the following recipe.

- Take a good  $[n, k]$ -error correcting code  $\mathcal{C} = \{c^1, c^2, \dots, c^{2^k}\} \subset \{0, 1\}^{1 \times n}$ .
- Let  $\mathcal{E}$  map “milestone” integers  $1 =: \mu_1 < \mu_2 < \dots < \mu_{2^k} := m$  to the codewords of  $\mathcal{C}$ , i.e.,  $\mathcal{E}(\mu_j) := c^j$ .
- “Interpolate” between the milestones. That is, if  $x \in [\mu_j, \mu_{j+1}]$ , then the prefix of  $\mathcal{E}(x)$  will come from  $\mathcal{E}(\mu_j)$  and the suffix from  $\mathcal{E}(\mu_{j+1})$ .

The technicality is with the third bullet point. A decoder of  $\mathcal{C}$  can translate  $\mathcal{E}(x)$  back to  $\mu_j$  if  $x$  is close enough to  $\mu_j$ . But there is going to be a middle ground between  $\mu_j$  and  $\mu_{j+1}$  such that the decoder will be confused.

To eliminate the confusion, Lolck and Pagh [6] proposed the following data structure

$$\mathcal{E}(\mu_j) := c^j \| c^j \| c^j \| c^j \in \{0, 1\}^{1 \times 4n},$$

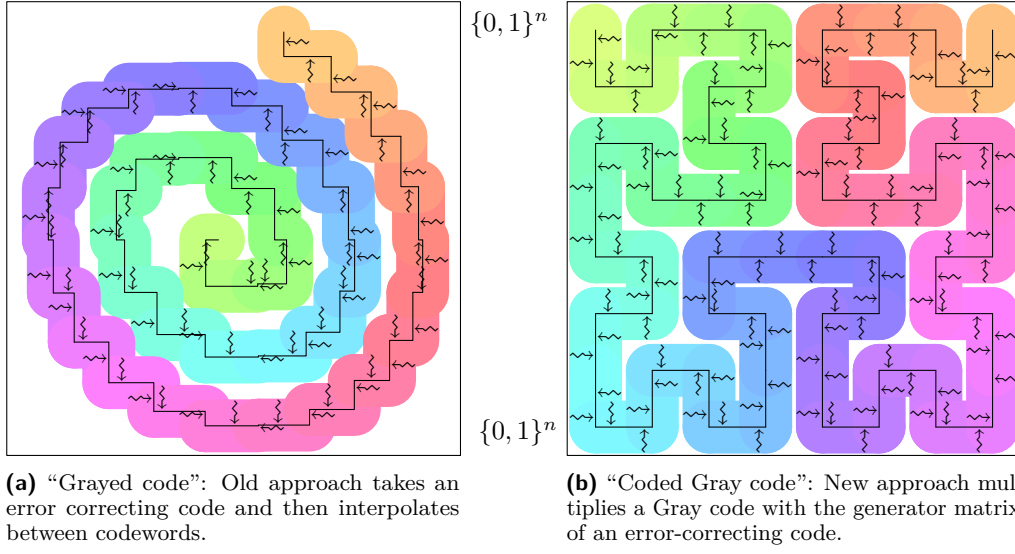
where  $\|$  is the string concatenation operator. They then interpolate between consecutive milestones  $\mu_j$  and  $\mu_{j+1}$  as

$$\begin{aligned} \mathcal{E}(\mu_{j.1}) &:= c^j \| c^j \| c^j \| c^j, \\ \mathcal{E}(\mu_{j.2}) &:= c^{j+1} \| c^j \| c^j \| c^j, \\ \mathcal{E}(\mu_{j.3}) &:= c^{j+1} \| c^{j+1} \| c^j \| c^j, \\ \mathcal{E}(\mu_{j.4}) &:= c^{j+1} \| c^{j+1} \| c^{j+1} \| c^j, \\ \mathcal{E}(\mu_{j.5}) &:= c^{j+1} \| c^{j+1} \| c^{j+1} \| c^{j+1} \end{aligned}$$

for some minor milestones  $\mu_j =: \mu_{j.1} < \mu_{j.2} < \mu_{j.3} < \mu_{j.4} < \mu_{j.5} := \mu_{j+1}$ . Note that only one copy is undergoing interpolation at any given time (which is highlighted). So the advantage of repeating  $c^j$  four times is that there are always two other copies that will decode to the same codeword. To elaborate, between  $\mu_{j.1}$  and  $\mu_{j.3}$ , the two  $c^j$  to the right will decode correctly; between  $\mu_{j.3}$  and  $\mu_{j.5}$ , the two  $c^{j+1}$  to the left will decode correctly.

Later, Fathollahi and Wootters [4] streamlined the data structure from  $4n$  bits to  $(2+3\epsilon)n$  bits by using *buffers* – consecutive zeros and ones. They map milestones to

$$\mathcal{E}(\mu_j) := 0^{\epsilon n} \| c^j \| 0^{\epsilon n} \| c^j \| 0^{\epsilon n} \in \{0, 1\}^{1 \times (2+3\epsilon)n}$$



■ **Figure 3** Old approach (not capacity-achieving) versus new approach (capacity-achieving).

if  $j$  is even, and to

$$\mathcal{E}(\mu_j) := 1^{\varepsilon n} \|c^j\| 1^{\varepsilon n} \|c^j\| 1^{\varepsilon n} \in \{0, 1\}^{1 \times (2+3\varepsilon)n}$$

if  $j$  is odd. They then interpolate between the milestones as

$$\begin{aligned} \mathcal{E}(\mu_{j.1}) &:= 0^{\varepsilon n} \|c^j\| 0^{\varepsilon n} \|c^j\| 0^{\varepsilon n}, \\ \mathcal{E}(\mu_{j.2}) &:= 1^{\varepsilon n} \|c^j\| 0^{\varepsilon n} \|c^j\| 0^{\varepsilon n}, \\ \mathcal{E}(\mu_{j.3}) &:= 1^{\varepsilon n} \|c^{j+1}\| 0^{\varepsilon n} \|c^j\| 0^{\varepsilon n}, \\ \mathcal{E}(\mu_{j.4}) &:= 1^{\varepsilon n} \|c^{j+1}\| 1^{\varepsilon n} \|c^j\| 0^{\varepsilon n}, \\ \mathcal{E}(\mu_{j.5}) &:= 1^{\varepsilon n} \|c^{j+1}\| 1^{\varepsilon n} \|c^{j+1}\| 0^{\varepsilon n}, \\ \mathcal{E}(\mu_{j.6}) &:= 1^{\varepsilon n} \|c^{j+1}\| 1^{\varepsilon n} \|c^{j+1}\| 1^{\varepsilon n} \end{aligned}$$

for some minor milestones  $\mu_j =: \mu_{j.1} < \mu_{j.2} < \mu_{j.3} < \mu_{j.4} < \mu_{j.5} < \mu_{j.6} =: \mu_{j+1}$ . In this construction, the decoder is left with two, not four, copies of  $c^j$ . It knows that the one sandwiched between  $0^{\varepsilon n}$  and  $1^{\varepsilon n}$  is the one undergoing interpolation, and hence the other one will decode correctly. To be more precise, between  $\mu_{j.1}$  and  $\mu_{j.4}$ , the left one is undergoing interpolation and the right  $c^j$  is trustworthy; between  $\mu_{j.3}$  and  $\mu_{j.6}$ , the right one is undergoing interpolation and the left  $c^{j+1}$  is trustworthy.

### 1.3 New approach

While this paper was in preparation, it came to our attention that Con, Fathollahi, Gabrys, Wootters, and Yaakobi have achieved similar results, but with different techniques [2]. In particular, their approach uses code concatenation.

In this and the concurrent work by Con, Fathollahi, Gabrys, Wootters, and Yaakobi, we aim to rightsize the length to  $n + \Theta(\varepsilon n)$  bits. While their work uses code concatenation, we begin with a generator matrix  $A \in \{0, 1\}^{k \times n}$  of some error-correcting code. We then reorder the codewords  $c^1, \dots, c^{2^k}$  using Gray code:

$$c^{j+1} = g^{j+1}A = (g^j + e^{\rho_j})A = c^j + A^{\rho_j} \in \{0, 1\}^{1 \times n}.$$

## 65:6 Capacity-Achieving Gray Codes

Here,  $g^j$  is the  $j$ th string of the Gray code,  $e^{\rho_j}$  is the  $\rho_j$ th cardinal vector, and  $A^{\rho_j}$  is the  $\rho_j$ th row of  $A$ , all as row vectors. Our data structure will look like

$$c^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}$$

or

$$c^j \| 1^{\varepsilon n} \| \rho_j \| \beta^j \| 1^{\varepsilon n} \| \rho_j \| \beta^j \| 1^{\varepsilon n}$$

depending on the parity of  $j$ . Here,  $\beta^j$  is a subvector of  $c^j$  obtained by collecting bits where  $A^{\rho_j}$  has 1. More precisely, if  $A^{\rho_j}$  has 1 at indices  $i_1, i_2, \dots, i_w$ , then  $\beta^j := c_{i_1}^j c_{i_2}^j \dots c_{i_w}^j \in \{0, 1\}^{1 \times w}$ , where  $w$  is the Hamming weight of  $A^{\rho_j}$ .

The purpose of keeping  $\rho_j$  in  $\mathcal{E}$  is to take note of which row of  $A$  we are going to add to  $c^j$  to obtain  $c^{j+1}$ . The purpose of keeping  $\beta^j$  in  $\mathcal{E}$  is to back up the bits of  $c^j$  that are going to be modified. We then interpolate between minor milestones

$$\begin{array}{l} c^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^j \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 0^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 1^{\varepsilon n} \| \rho_j \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^j \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 0^{\varepsilon n}, \\ c^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 1^{\varepsilon n} \| \rho_{j+1} \| \beta^{j+1} \| 1^{\varepsilon n}, \end{array}$$

Note that we use Fathollahi and Wootters's data structure to protect  $\rho_j$  and  $\beta^j$ , and so the backup data is almost always available.<sup>4</sup>

An analogy of this construction is to think of  $c_j$  as the state of our computer at version  $j$ . Now a software update comes in and attempts to add  $A^{\rho_j}$  to  $c_j$ . To avoid messing things up, the updater backs up the files that are going to be updated, which are at  $i_1, \dots, i_w$ ; and  $\beta^j$  is the backup data.

Our construction, at any code rate below capacity, achieves positive error exponents and linear encoding and decoding complexity.

► **Theorem 1 (Main theorem).** *Fix a BSC with  $p \in (0, 1/2)$  and a gap to capacity  $\varepsilon > 0$ . For sufficiently large  $n$ , there exists a pair of encoder  $\mathcal{E}: [m] \rightarrow \{0, 1\}^{1 \times n}$  and decoder  $\mathcal{D}: \{0, 1\}^{1 \times n} \rightarrow [m]$  with code size  $m > 2^{(\text{Capacity}(\text{BSC}_p) - \varepsilon)n}$  such that (a)  $\mathcal{E}(x)$  and  $\mathcal{E}(x + 1)$  differ by one bit and (b)*

$$\text{Prob} \left\{ \left| \mathcal{D}(\text{BSC}_p^n(\mathcal{E}(x))) - x \right| > t \right\} < 2^{-\Omega(n)} + 2^{-\Omega(t)}$$

for all  $x \in [m - 1]$  and all  $t > 1$ . Moreover, the time complexity of  $\mathcal{E}$  and  $\mathcal{D}$  scales<sup>5</sup> linearly in  $n$ .

### Organization

The rest of the paper is dedicated to proving Theorem 1.

<sup>4</sup> There is no reason not to protect  $\rho_j$  and  $\beta^j$  using error-correcting codes. We omit that here but will discuss in the formal proof.

<sup>5</sup> exponentially in  $1/\text{poly}(\varepsilon)$



## 2 Proof of Theorem 1

Fix a crossover probability  $p$  and a gap to capacity  $\varepsilon > 0$ . Let  $n$  and  $k$  be very large.

### 2.1 The building blocks $\mathcal{B}$ and $\mathcal{C}$

We begin with a linear code  $\mathcal{B}$  with block length  $\varepsilon n$  and dimension  $\varepsilon k$ . Using well-known constructions [5, Theorem 8], we can make the code rate  $k/n$   $\varepsilon$ -close to the capacity if  $n$  is large enough. Moreover, the encoding and decoding complexity can be made linear in  $n$ . Let  $B$  be the generator matrix of  $\mathcal{B}$ .

We stack  $B$  to construct a larger generator matrix

$$A := \begin{bmatrix} B & \bar{B} & & & & \\ & B & \bar{B} & & & \\ & & B & \bar{B} & & \\ & & & \ddots & \ddots & \\ & & & & B & \bar{B} \end{bmatrix} \in \{0, 1\}^{k \times (1+\varepsilon)n} \quad (4)$$

and denote the corresponding code by  $\mathcal{C} \subset \{0, 1\}^{1 \times (1+\varepsilon)n}$ . Here,  $\bar{B}$  is the bitwise complement of  $B$ . We put  $\bar{B}$  next to  $B$  so that all rows of  $[B \ \bar{B}]$  has the same Hamming weight,  $\varepsilon n$ . This means that the backup data  $\beta^j$  will be exactly  $\varepsilon n$  bits long. We repeat  $B$  and  $\bar{B}$   $1/\varepsilon$  times so that  $A$  has block length  $(1 + \varepsilon)n$  and dimension  $k$ . This makes the code rate of  $\mathcal{C}$   $2\varepsilon$ -close to the capacity.

Decoding  $\mathcal{C}$  is straightforward. Given a received word  $y \in \{0, 1\}^{1 \times (1+\varepsilon)n}$ , apply  $\mathcal{B}$ 's decoder to  $y_1, \dots, y_{\varepsilon n}$  to obtain  $x_1, \dots, x_{\varepsilon n}$ . Subtract the influence of  $x_1, \dots, x_{\varepsilon n}$  from  $y_{\varepsilon n+1}, \dots, y_{2\varepsilon n}$  and apply  $\mathcal{B}$ 's decoder to obtain  $x_{\varepsilon n+1}, \dots, x_{2\varepsilon n}$ . Repeat this process until we obtain  $x_n$ .

We also use  $\mathcal{B}$  to protect the row index  $\rho_j \in [k]$  and the backup data  $\beta^j \in \{0, 1\}^{\varepsilon n}$ . Denote by  $\mathcal{B}(\rho_j, \beta^j)$  the result of encoding these  $\log_2(k) + \varepsilon n$  bits of information using

$$\frac{2n}{k} \geq \left\lceil \frac{\log_2(k) + \varepsilon n}{\varepsilon k} \right\rceil$$

blocks of  $\mathcal{B}$ . This means that  $\mathcal{B}(\rho_j, \beta^j)$  has length  $2\varepsilon n^2/k$ .

### 2.2 Encoding $\mathcal{E}$

Recall that  $\rho_j$  is the ruler sequence defined in (2) capped at  $k$ . Recall that  $g^j$  is the  $j$ th string of the Gray code and is obtained by flipping the  $\min(\rho_{j-1}, k)$ th bit of  $g^{j-1}$ . We assume an ordering on the codewords  $\mathcal{C} = \{c^1, \dots, c^{2^k}\}$  by Gray code, i.e.,  $c^j := g^j A$ . Let  $\beta^j$  be the subvector of  $c^j$  obtained by deleting the bits where  $A^{\rho_j}$  has 0.

We now place the milestones at

$$\mu_j := j\varepsilon n(4 + 2n/k)$$

for  $j \in [2^k]$ . Consequently,  $\mathcal{E}$  will encode integers up to  $m := (2^k - 1)\varepsilon n(4 + 2n/k) + 1 = (1 + o(1))2^k$ . We then define data structure:

$$\mathcal{E}(\mu_j) := c^j \|0^{\varepsilon n} \| \mathcal{B}(\rho_j, \beta^j) \| 0^{\varepsilon n} \| \mathcal{B}(\rho_j, \beta^j) \| 0^{\varepsilon n}$$

and

$$\mathcal{E}(\mu_j) := c^j \|1^{\varepsilon n} \| \mathcal{B}(\rho_j, \beta^j) \| 1^{\varepsilon n} \| \overline{\mathcal{B}(\rho_j, \beta^j)} \| 1^{\varepsilon n}$$

depending on the parity of  $j$ . Here,  $\overline{\mathcal{B}(\rho_j, \beta^j)}$  is the bitwise complement of  $\mathcal{B}(\rho_j, \beta^j)$ . Note that each  $\mathcal{E}(\mu_j)$  is  $(1 + 4\varepsilon + 4\varepsilon n/k)n$  bits long. We infer that the code rate of  $\mathcal{E}$  is  $\mathcal{O}(\varepsilon)$ -close to the capacity.

Next, we show that the Hamming distance between  $\mathcal{E}(\mu_j)$  and  $\mathcal{E}(\mu_{j+1})$  is  $\varepsilon n(4 + 2n/k)$ . Trivially, the consecutive zeros and ones contributes  $3\varepsilon n$  bits of Hamming distance. Next, note that

$$c^{j+1} - c^j = g^{j+1}A - g^jA = (g^{j+1} - g^j)A = e^{\rho_j}A = A^{\rho_j}.$$

This is the  $\rho_j$ th row of  $A$ . By the construction (4), any row of  $A$  contributes exactly  $\varepsilon n$  bits of Hamming distance. Next,  $\mathcal{B}(\rho_j, \beta^j)$  and  $\mathcal{B}(\rho_{j+1}, \beta^{j+1})$  contributes an unknown amount of distance. But it is complement to the distance between  $\mathcal{B}(\rho_j, \beta^j)$  and  $\overline{\mathcal{B}(\rho_{j+1}, \beta^{j+1})}$ . Therefore, the  $\mathcal{B}$  part contributes exactly  $2\varepsilon n^2/k$ . In total, the Hamming distance is exactly  $\varepsilon n(4 + 2n/k)$ .

Now that the distance between consecutive milestones matches the Hamming distance, we can interpolate between them. In particular, for even  $j$ ,

$$\begin{aligned} \mathcal{E}(\mu_j) &:= c^j \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + \varepsilon n) &:= c^{j+1} \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + 2\varepsilon n) &:= c^{j+1} \|1^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + 2\varepsilon n + d) &:= c^{j+1} \|1^{\varepsilon n}\| \mathcal{B}(\rho_{j+1}, \beta^{j+1}) \|0^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + 3\varepsilon n + d) &:= c^{j+1} \|1^{\varepsilon n}\| \mathcal{B}(\rho_{j+1}, \beta^{j+1}) \|1^{\varepsilon n}\| \mathcal{B}(\rho_j, \beta^j) \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + 3\varepsilon n + 2\varepsilon n^2/k) &:= c^{j+1} \|1^{\varepsilon n}\| \mathcal{B}(\rho_{j+1}, \beta^{j+1}) \|1^{\varepsilon n}\| \overline{\mathcal{B}(\rho_{j+1}, \beta^{j+1})} \|0^{\varepsilon n}\|, \\ \mathcal{E}(\mu_j + 4\varepsilon n + 2\varepsilon n^2/k) &:= c^{j+1} \|1^{\varepsilon n}\| \mathcal{B}(\rho_{j+1}, \beta^{j+1}) \|1^{\varepsilon n}\| \overline{\mathcal{B}(\rho_{j+1}, \beta^{j+1})} \|1^{\varepsilon n}\|, \end{aligned}$$

where  $d$  is the Hamming distance between  $\mathcal{B}(\rho_j, \beta^j)$  and  $\mathcal{B}(\rho_{j+1}, \beta^{j+1})$ .

## 2.3 Decoding $\mathcal{D}$

Suppose that we are given

$$c \| \phi \| B' \| \phi' \| B'' \| \phi''' \tag{5}$$

as the noisy version of  $\mathcal{E}(x)$  for some  $x \in [m]$ , where

- $c \in \{0, 1\}^{1 \times (1+\varepsilon)n}$  is the noisy version of  $c^j$ ,  $c^{j+1}$ , or anything in between,
- $\phi, \phi', \phi'' \in \{0, 1\}^{1 \times \varepsilon n}$  are the noisy version of the buffers, and
- $B', B'' \in \{0, 1\}^{1 \times 2\varepsilon n^2/k}$  are the noisy version of the  $\mathcal{B}$  part.

We first apply Fathollahi and Wootters's decoder [4] to the second half of (5)

$$\phi \| B' \| \phi' \| B'' \| \phi'''.$$

Their decoder counts how many ones and zeros are in  $\phi$ ,  $\phi'$ , and  $\phi''$ . This tells us which minor milestone we are at. We use this information to determine which of  $B'$  or  $B''$  is undergoing interpolation, and which is trustworthy. And then, we use the trustworthy one to recover the row index  $\rho_j$  and the backup data  $\beta^j$  (or  $\rho_{j+1}$  and  $\beta^{j+1}$  depending on if  $x$  is past  $\mu_j + 2.5\varepsilon n + d$  or not). From now on, we just call them  $\rho$  and  $\beta$ .

We define the rollback function  $\text{Roll}: \{0, 1\}^{(1+\varepsilon)n} \times [k] \times \{0, 1\}^{\varepsilon n} \rightarrow \{0, 1\}^{(1+\varepsilon)n}$  that overwrites messed-up bits using backup data. More precisely,  $\text{Roll}(c, \rho, \beta)$  will be the vector  $c$  after replacing  $c_{i_1}$  with  $\beta_1$ ,  $c_{i_2}$  with  $\beta_2$ , and so on, where  $i_1, i_2, \dots$  are the indices where  $A^\rho$  has 1. Our claim is that, it does not matter if it is  $c^j$ ,  $0^{\varepsilon n}$ , or  $\mathcal{B}$  that is undergoing interpolation,  $\text{Roll}(c, \rho, \beta)$  will just look like the noisy version of  $c^j$  or  $c^{j+1}$ , which can be decoded by the decoder of  $\mathcal{C}$ . This can be seen more clearly by considering three cases.

Case 1:  $x$  is between  $\mu_j$  and  $\mu_j + \varepsilon n$ . This is the stage where we are adding  $A^p$  to  $c^j$ . In this case, the  $\mathcal{B}$  part backs up the subvector of  $c^j$  that is undergoing interpolation;  $\text{Roll}(c, \rho, \beta)$  would just be a noisy version of  $c^j$  that can be decoded by  $\mathcal{C}$ .

Case 2:  $x$  is between  $\mu_j + \varepsilon n$  and  $\mu_j + 2.5\varepsilon n + d$ . This is the case where  $B'$  is not trustworthy and so Fathollahi and Wootters's decoder will decode  $B''$  to  $(\rho_j, \beta^j)$ . In this case,  $\text{Roll}(c, \rho_j, \beta^j)$  will be a noisy version of  $c^j$  that can be decoded by  $\mathcal{C}$ .

Case 3:  $x$  is between  $\mu_j + 2.5\varepsilon n + d$  and  $\mu_j + 4\varepsilon n + 2\varepsilon n^2/k$ . This is the case where  $B''$  is not trustworthy and so Fathollahi and Wootters's decoder will decode  $B'$  to  $\rho_{j+1}, \beta^{j+1}$ . In this case,  $\text{Roll}(c, \rho_{j+1}, \beta^{j+1})$  will be a noisy version of  $c^{j+1}$  that can be decoded by  $\mathcal{C}$ .

Examining these three cases, we can see that  $\text{Roll}(c, \rho, \beta)$  will always yield  $c^j$  or  $c^{j+1}$ . With that, we can compute  $g^j$  and  $j$ . Now that we know  $x \in [\mu_j, \mu_{j+1}]$ , it suffices to compare (5) with  $\mathcal{E}(\mu_j), \dots, \mathcal{E}(\mu_{j+1})$  and see which one minimizes the Hamming distance. The minimizer will be our best bet of  $x$ .

## 2.4 Complexity and tail estimation

The complexity of  $\mathcal{E}$  and  $\mathcal{D}$  is linear in  $n$ . This is because Gray's encoding, Gray's decoding,  $\mathcal{B}$ 's encoding,  $\mathcal{B}$ 's decoding, determining whether  $\phi, \phi'$ , and  $\phi''$  are zeros or ones, determining whether  $B'$  or  $B''$  is trustworthy, and Roll are all linear in  $n$ .

The tail estimation (3) boils down to the following components.

- With probability  $2^{-\Omega(n)}$ , we obtain the wrong  $j$ , i.e.,  $x \notin [\mu_j, \mu_{j+1}]$ .
- The guesswork of  $x$  conditioned on correct  $j$  has tail probability  $2^{-\Omega(t)}$ .

The first bullet point is a consequence of the error probability of  $\mathcal{B}$  being  $2^{-\Omega(\varepsilon n)}$ , which is  $2^{-\Omega(n)}$  as we fixed  $\varepsilon$ . The second bullet point relies on what minimizes the Hamming distance between (5) and  $\mathcal{E}(\mu_j), \dots, \mathcal{E}(\mu_{j+1})$ . Such analysis has been done before [6, Lemma 3.7] [4, Lemma 13], and we do not repeat it here. This finishes the proof.

---

## References

- 1 Martin Aumüller, Christian Janos Lebeda, and Rasmus Pagh. Representing Sparse Vectors with Differential Privacy, Low Error, Optimal Space, and Fast Access. *Journal of Privacy and Confidentiality*, 12(2), November 2022. doi:10.29012/jpc.809.
- 2 Roni Con, Dorsa Fathollahi, Ryan Gabrys, Mary Wootters, and Eitan Yaakobi. Robust gray codes approaching the optimal rate, 2024. arXiv:2406.17689.
- 3 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51, 2016. doi:10.29012/jpc.v7i3.405.
- 4 Dorsa Fathollahi and Mary Wootters. Improved construction of robust gray code, 2024. arXiv:2401.15291.
- 5 Venkatesan Guruswami and Piotr Indyk. Linear-Time Encodable/Decodable Codes With Near-Optimal Rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, October 2005. doi:10.1109/TIT.2005.855587.
- 6 David Rasmussen Lolck and Rasmus Pagh. Shannon meets Gray: Noise-robust, Low-sensitivity Codes with Applications in Differential Privacy. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 1050–1066. Society for Industrial and Applied Mathematics, January 2024. doi:10.1137/1.9781611977912.40.
- 7 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2024. Published electronically at <http://oeis.org>.



# On Black-Box Meta Complexity and Function Inversion

Noam Mazon ✉

Tel Aviv University, Israel

Rafael Pass ✉

Tel Aviv University, Israel

Cornell Tech, New York, NY, USA

---

## Abstract

---

The relationships between various meta-complexity problems are not well understood in the *worst-case regime*, including whether the search version is harder than the decision version, whether the hardness scales with the “threshold”, and how the hardness of different meta-complexity problems relate to one another, and to the task of function inversion.

In this work, we present resolutions to some of these questions with respect to the *black-box* analog of these problems. In more detail, let  $\text{MK}_M^t P[s]$  denote the language consisting of strings  $x$  with  $K_M^t(x) < s(|x|)$ , where  $K_M^t(x)$  denotes the  $t$ -bounded Kolmogorov complexity of  $x$  with  $M$  as the underlying (Universal) Turing machine, and let  $\text{search-MK}_M^t P[s]$  denote the search version of the same problem.

We show that if for *every* Universal Turing machine  $U$  there exists a  $2^{\alpha n} \text{poly}(n)$ -size  $U$ -oracle aided circuit deciding  $\text{MK}_U^t P[n - O(1)]$ , then for every function  $s$ , and every *not necessarily universal* Turing machine  $M$ , there exists a  $2^{\alpha s(n)} \text{poly}(n)$ -size  $M$ -oracle aided circuit solving  $\text{search-MK}_M^t P[s(n)]$ ; this in turn yields circuits of roughly the same size for both the Minimum Circuit Size Problem (MCSP), and the function inversion problem, as they can be thought of as instantiating  $\text{MK}_M^t P$  with particular choices of (a non-universal) TMs  $M$  (the circuit emulator for the case of MCSP, and the function evaluation in the case of function inversion).

As a corollary of independent interest, we get that the complexity of black-box function inversion is (roughly) the same as the complexity of black-box deciding  $\text{MK}_U^t P[n - O(1)]$  for any universal TM  $U$ ; that is, also in the worst-case regime, black-box function inversion is “equivalent” to black-box deciding  $\text{MK}_U^t P$ .

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

**Keywords and phrases** Meta Complexity, Kolmogorov complexity, function inversion

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.66

**Category** RANDOM

**Related Version** *Full Version:* <https://eprint.iacr.org/2023/1832.pdf>

**Funding** *Noam Mazon:* Research partly supported by NSF CNS-2149305 and DARPA under Agreement No. HR00110C0086.

*Rafael Pass:* Supported in part by AFOSR Award FA9550-23-1-0387, AFOSR Award FA9550-23-1-0312, and an Algorand Foundation grant. This material is based upon work supported by DARPA under Agreement No. HR00110C0086. 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 United States Government, DARPA, AFOSR or the Algorand Foundation.

## 1 Introduction

We consider the worst-case complexity of solving standard meta-complexity Programs, notably the the *Time-Bounded Kolmogorov Complexity Problem* [10, 19, 1, 9, 5, 18] – computing the length, denoted  $K_U^t(x)$  of the shortest program (evaluated on some particular Universal



© Noam Mazon and Rafael Pass;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 66; pp. 66:1–66:12



Leibniz International Proceedings in Informatics

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

Turing machine (TM)  $U$  that generates a given string  $x$ , within time  $t(|x|)$ , where  $t$  is a polynomial, and the (b) the *Minimum Circuit Size problem* (MCSP) [8, 20] – finding the smallest Boolean circuit that computes a given function  $x$ . For both of these problem one may also consider *thresholds* versions,  $\text{MK}_{\cup}^t\text{P}[s]$  and  $\text{MCSP}[s]$ , where  $\text{MK}_{\cup}^t\text{P}[s]$  (resp.  $\text{MCSP}[s]$ ) is the languages of strings  $x$  s.t.  $K_{\cup}^t(x)$  (resp. the circuit size of  $x$ ) is less than  $s(|x|)$ , as well as *search* versions  $\text{search-MK}_{\cup}^t\text{P}[s]$ , where the goal is not only to compute/decide the complexity of a string  $x$  but also to *find* a short description that witnesses this complexity.

The relationship between these various meta-complexity problems are not well understood. In particular:

1. **Decision-to-Search:** Solving the search version trivially yields a solver for the decisional (or computational) version with roughly the same complexity. Does the converse hold: Does a  $T(n)$ -size circuit for solving the decision version imply a, roughly,  $T(n)$ -size circuit solving the search version?
2. **Hardness Scaling to the Threshold:** Intuitively, the threshold version of the problem, for small thresholds  $s(n) \ll n$  ought to be easier than the threshold  $n$  version (or computational) version since there exist trivial  $2^{s(n)}\text{poly}(n)$  time algorithms for the threshold version (simply doing brute force search). Does this hold more generally: Does a  $T(n)$ -size circuit solving  $\text{MK}_{\cup}^t\text{P}[n - O(1)]$  imply a roughly  $T(s(n))$ -size circuit for solving  $\text{MK}_{\cup}^t\text{P}[s(n)]$ ?
3. **The “Model of Computation” and the Relationship to Function Inversion:** Other meta-complexity problems, such as the MCSP problem, can be stated as an  $\text{MK}_M^t\text{P}$  problem with respect to a particular *non-universal* underlying TM  $M$  (performing circuit emulation). Additionally, a solver for the search- $\text{MK}_M^t\text{P}$  problem with respect to *any* (non-universal) TM  $M$  is also equivalent to a solver for the function inversion problem (i.e., the problem of inverting any function on every input). Does a  $T(n)$ -size solver for  $\text{MK}_{\cup}^t\text{P}[s(n)]$  with respect to any underlying Universal TM  $U$ , imply one (of size roughly  $T(n)$ ) that also works with respect non-universal TMs (and thus also for MCSP and function inversion)?

In the *average-case regime*, positive answers to these questions – when restricting to efficient underlying (Universal) TMs – were provided in respectively [11] (for question 1), [12] (for question 2) and [11, 17] (for question 3), but they remain wide open in the worst-case regime. This is the focus of the current paper, but rather than restricting to efficient underlying TMs, we will consider *arbitrary* TMs (with potentially a large description or running time).

In particular, very recently non-trivial circuits for the various different meta-complexity problems were given. In [14], the current authors show that for any efficient Universal TM  $U$ , there exists a circuit of size  $2^{4n/5}\text{poly}(n, t(n))$  that solves the search version of the  $K_{\cup}^t$  (and thus also search- $\text{MK}_{\cup}^t\text{P}$ ). A different, independent, paper by Hirahara, Ilango and Williams [6] focuses on the threshold version of the above meta-complexity problems and presents circuits of size respectively  $2^{(4/5)s(n)} \cdot \text{poly}(n, t(n))$  and  $2^{(4/5+o(1)) \cdot s(n) \log s(n)}$  for them. In both cases, the core of the technical work consist of providing a circuit implementation of the function inversion algorithm from Fiat and Naor [3], and next applying this function inversion algorithm to the one-way function construction of [11] (or variants thereof, notably the variant of [17] to deal with the MCSP problem) based on the hardness of meta-complexity problems – an approach first envisioned by Ren and Santhanam [17].<sup>1</sup> As such, the worst-case

<sup>1</sup> [17] noted that the function inversion algorithm of [3] could be applied to the one-way function construction of [11] to get a non-trivial non-uniform RAM program that solves the  $\text{MK}^t\text{P}$  problem, but left open whether a circuit implementation can be given.

complexity bounds obtained are roughly the same for (a) the search and the decisional version (as the function inverter also directly solves the search problem in [11]), (b) they naturally scale with the threshold  $s$  of  $\text{MK}^t\text{P}[s]$  (based on an extension of the function inversion attack of Fiat-Naor done in [6]), and (c) are the roughly the same for  $\text{MK}^t\text{P}$ ,  $\text{MCSP}$  and function inversion (since the one-way function constructions in [11, 17] are length preserving). These works thus indicate that perhaps the same phenomena that are known in the average-case setting may also hold in the worst-case setting.

In this paper, we demonstrate that this is not a coincidence. Indeed, we provide a positive answer to all the above questions also in the worst-case regime, when restricting attention to *black-box* solvers and thus all the above result follow from simply obtaining a circuit for black-box solving the decisional  $\text{MK}^t\text{P}$  problem.

### Black-box Solvers

As noted in [14], their algorithm for  $K_{\cup}^t$  works for any Universal TM  $U$  (as long as the algorithm gets oracle access to  $U$ ): for *any* (not necessarily efficient) Universal TM  $U$ , there exists a  $U$ -oracle aided circuit of size  $2^{4n/5}\text{poly}(n, t(n))$  that solves the search version of the  $K_{\cup}^t$ . Following [14], we say that  $\text{MK}^t\text{P}[s]$  (resp search-  $\text{MK}^t\text{P}[s]$ ) *admits a  $T(n)$ -size black-box solver* if for every universal TM  $U$ , there exists a  $T(n)$  size  $U$ -oracle aided circuit for solving  $\text{MK}_{\cup}^t\text{P}[s]$  (resp search-  $\text{MK}_{\cup}^t\text{P}[s]$ ). We additionally say that these problems admit a  $T(n)$ -size *generalized black-box solver* if the same holds not only with respect to any *universal* TM  $U$  but also for *non-universal* TMs  $M$  (satisfying the minimal condition that the emulation by  $M$  has a unique output:  $M(\Pi, 1^{t_1}) = M(\Pi, 1^{t_2})$  if either of those provide some output). Considering generalized black-box solvers is what will allow us to answer question 3 above, but actually, also from a technical point of view, will also be instrumental also to deal with question 1.

[14] proved a lower bound of  $2^{n/2-o(n)}$  on the size of black-box  $K^t$  solvers.

## 1.1 Our Results

Our main result shows that the existence of a  $2^{\alpha n+o(n)}$ -size black-box solver for  $\text{MK}^t\text{P}[n - O(1)]$  implies the existence of a  $2^{\alpha s(n)+o(n)}$ -size *generalized* black-box solver for search-  $\text{MK}^t\text{P}[s]$ , thus providing a positive answer to all the above questions with respect to black-box solvers.

► **Theorem 1.** *Assume the existence of a  $2^{\alpha n} \cdot \text{poly}(n)$ -size black-box solver for  $\text{MK}^t\text{P}[n - 4]$  for  $t(n) = n$ . Then there exists a  $2^{\alpha s(n)} \cdot \text{poly}(n)$ -size generalized black-box solver for search-  $\text{MK}^t\text{P}[s]$  for every function  $t'(\cdot)$  and every function  $s(n) \leq 2n - \lceil \log n \rceil$ .*

We highlight that generalized black-box solvers can solve  $\text{MCSP}$  (since, as implicitly observed in [6] following [17, 4]), the  $\text{MCSP}$  problem can be stated as an  $\text{MK}_{\mathcal{M}}^t\text{P}$  problem with respect to a particular *non-universal* underlying TM  $M$  (performing circuit emulation) – see Lemma 19 in the Appendix). As a direct corollary of Theorem 1 and Lemma 19, we thus get:<sup>2</sup>

► **Corollary 2.** *Let  $p \in \text{poly}$  and  $\alpha > 0$ , and assume that for  $t(n) = n$  there exists a black-box  $\text{MK}^t\text{P}[n - 4]$  solver of size  $2^{\alpha n} \cdot p(n)$ . Then for every  $s(n) \leq 1.9n/\log n$ , search-  $\text{MCSP}[s]$  can be solved with a circuit family of size  $2^{(\alpha+o(1)) \cdot s(n) \cdot \log(s(n)+\log n)} \cdot \text{poly}(n)$ .*

<sup>2</sup> When  $s(n) \geq 1.1 \cdot n/\log n$  then  $\text{MCSP}[s]$  is the trivial language consisting of all strings due to the result of [13], so the corollary below actually works for all  $s$ .

Additionally, we observe that generalized black-box solvers for search- $\text{MK}^{\text{tP}}[n]$  can easily be seen to be equivalent to function inversion circuits (for all functions  $f$ ) of roughly the same size – see Lemmas 16 and 17 in the Appendix. As a corollary of Theorem 1, we thus get that – in the black-box regime – solving the function inversion problem is not only sufficient (as shown in [14, 6] for solving  $\text{MK}^{\text{tP}}[n - O(1)]$ ) but also *necessary*. This matches the converse direction of the *average-case* characterization of one-way functions through the hardness of  $\text{MK}^{\text{tP}}$  from [11], and yields a characterization of the *black-box* worst-case hardness of  $\text{MK}^{\text{tP}}$  through the black-box worst-case hardness of one-way functions. In particular, black-box solving just  $\text{MK}^{\text{tP}}[n - O(1)]$  is no easier than (black-box) function inversion.

► **Theorem 3.** *There exists a black-box  $\text{MK}^{\text{tP}}[n - O(1)]$  solver of size  $2^{\alpha n} \cdot \text{poly}(n)$  for every polynomial  $t$  if and only if every function  $f$  can be inverted by an  $f$ -oracle aided circuit of size  $2^{\alpha n} \cdot \text{poly}(n)$ .*

As a consequence of Theorem 3, and Impagliazzo’s lower bound on the circuit size of black-box one-way function inversion [7]<sup>3</sup>, we directly get a lower bound on the complexity of black-box  $\text{MK}^{\text{tP}}$  solvers; such a lower bound was previously proved directly for the  $\text{MK}^{\text{tP}}$  problem in [14] but it required a significantly more complicated proof and employing heavier machinery.

► **Corollary 4.** *There is no black-box  $\text{MK}^{\text{tP}}[n - 4]$  solver of size  $2^{n/2 - o(n)}$ .*

## 1.2 Proof Outline

Theorem 1 is proved in two steps. The first step is formally stated in Corollary 13 and the second in Corollary 15.

### Step 1: From Black-box to Generalized Black-Box for Small Thresholds

We first show that any black-box solver for  $\text{MK}^{\text{tP}}[n - 4]$  of size  $T(n)$  implies a generalized black box  $\text{MK}^{\text{tP}}[s(n)]$  solver of size  $T(s(n) + O(1))\text{poly}(n, t(n))$ .<sup>4</sup> The proof follows standard techniques from the literature on hardness magnification (i.e., hashing down the statement  $x$  using a pairwise independent hash function  $h$  to roughly the threshold size, and then applying the solver of a related language on the smaller instance  $h(x)$  and thereby improving the running time) [16, 2, 15]. The key difference with our approach is that by leveraging the black-box property of the algorithm, we can use an algorithm for the *same* problem, but parameterized by a different universal TM  $M_h$ , as opposed to a general **NP** problem as in those earlier works – that is, we get “self hardness magnification” [12]). (We highlight that [6] also rely on a similar hashing technique to *directly* present an attack on the threshold version of  $\text{MK}^{\text{tP}}$  but do so in a slightly different context: in particular, they use hashing to develop a function inversion algorithm whose circuit complexity only depends on the input size of the function and not the output size, and next function inversion with an input size that depends on the threshold to solve  $\text{MK}^{\text{tP}}[s]$ . Nevertheless, our usage of this approach is inspired by theirs.) Additionally, and perhaps more surprisingly, we show that this technique allows us to solve the orthogonal problem of dealing with *non-universal* Turing machines (so that we can get a generalized black-box solver): in essence, the idea is to define a universal

<sup>3</sup> Impagliazzo shows that for every large enough  $n \in \mathbb{N}$ , there exists a permutation  $\sigma: \{0, 1\}^n \rightarrow \{0, 1\}^n$  such that every  $\sigma$ -oracle aided circuit  $C$  of size at most  $2^{n/2 - 2 \log^2 n}$  fails to invert  $f$ . See also [21].

<sup>4</sup> This reduction also works for the search version of these problems.



TM  $M_h$  that has two tracks: if the first bit of the input “program”  $\Pi$  is 0, it simply runs some Universal TM  $U(\Pi_{>1})$  on the rest of the input  $\Pi_{>1}$ , and if it is 1, then it outputs  $h(M(\Pi_{>1}))$  where  $M$  is the non-universal TM that we want a  $\text{MK}_M^t\text{P}[s]$  solver for. The key point is that due to pairwise independence property of the hash function,  $h(x)$  is uniform (for a random choice of  $h$ ) and thus with high probability  $h(x)$  has essentially maximal  $K_U^t$  complexity, and thus the existence of the first “track” does not disrupt the hardness magnification reduction.

## Step 2: From Decision to Search

Our next result shows how any generalized  $\text{MK}^t\text{P}[s]$  solver of size  $2^{\alpha s(n)}$  can be used to solve also the *search version* of the problem with roughly the same running time. In particular, to solve search- $\text{MK}_M^t\text{P}[s]$ , we will rely on a circuit deciding  $\text{MK}_{M_n}^t\text{P}[s + \lceil \log n \rceil]$  where  $M_n$  is defined as a TM that given a program  $\Pi = (i, \Pi')$  where  $i$  is defined as the first  $\lceil \log n \rceil$  bits of  $\Pi$ , checks if  $\Pi'$  generates an output  $x$  of exactly  $n$  bits, and if so outputs  $x$  concatenated with the first  $i$  bits of  $\Pi'$ . The key point is that for every  $n$ -bit length string  $x$ ,  $K_{M_n}^t(x) = K_M^t(x) + \lceil \log n \rceil$  (obtained by letting  $i = 0$ ). Furthermore, this Kolmogorov complexity can be maintained if we concatenate the prefix of any minimum length program  $\Pi'$  that generates  $x$ , so the bits of any such minimum length program can be iteratively recovered given an oracle computing  $K_{M_n}^t$ . The same argument also works if we only have access to a decision oracle for the threshold  $s + \lceil \log n \rceil$ , but then we only recover a program of length at most  $s$ .

## 2 Preliminaries

### 2.1 Notations

All logarithms are taken in base 2. We use calligraphic letters to denote sets and distributions, uppercase for random variables, and lowercase for values and functions. Let  $\text{poly}$  stand for the set of all polynomials. Given a vector  $v \in \Sigma^n$ , let  $v_i$  denote its  $i^{\text{th}}$  entry, let  $v_{<i} = (v_1, \dots, v_{i-1})$  and  $v_{\leq i} = (v_1, \dots, v_i)$ . Similarly, for a set  $\mathcal{I} \subseteq [n]$ , let  $v_{\mathcal{I}}$  be the ordered sequence  $(v_i)_{i \in \mathcal{I}}$ . For a function  $f: \mathcal{D} \rightarrow \mathcal{R}$ , and a set  $\mathcal{S} \subseteq \mathcal{D}$ , we let  $f(\mathcal{S}) = \{f(x) : x \in \mathcal{S}\}$ .

### 2.2 Distributions and Random Variables

When unambiguous, we will naturally view a random variable as its marginal distribution. The support of a finite distribution  $\mathcal{P}$  is defined by  $\text{Supp}(\mathcal{P}) := \{x : \Pr_{\mathcal{P}}[x] > 0\}$ . For a (discrete) distribution  $\mathcal{P}$ , let  $x \leftarrow \mathcal{P}$  denote that  $x$  was sampled according to  $\mathcal{P}$ . Similarly, for a set  $\mathcal{S}$ , let  $x \leftarrow \mathcal{S}$  denote that  $x$  is drawn uniformly from  $\mathcal{S}$ .

### 2.3 Kolmogorov Complexity

Roughly speaking, the *t-time-bounded Kolmogorov complexity*,  $K^t(x)$ , of a string  $x \in \{0, 1\}^*$  is the length of the shortest program  $\Pi = (M, y)$  such that, when simulated by an universal Turing machine,  $\Pi$  outputs  $x$  in  $t(|x|)$  steps. Here, a program  $\Pi$  is simply a pair of a Turing Machine  $M$  and an input  $y$ , where the output of  $P$  is defined as the output of  $M(y)$ . When there is no running time bound (i.e., the program can run in an arbitrary number of steps), we obtain the notion of Kolmogorov complexity.

In the following, let  $U(\Pi, 1^t)$  denote the output of  $\Pi$  when emulated on  $U$  for  $t$  steps. We now define the notion of Kolmogorov complexity with respect to the universal TM  $U$ .

## 66:6 On Black-Box Meta Complexity and Function Inversion

► **Definition 5.** Let  $t$  be a polynomial. For all  $x \in \{0, 1\}^*$ , define

$$K_{\mathcal{U}}^t(x) = \min_{\Pi \in \{0, 1\}^*} \{|\Pi| : \mathcal{U}(\Pi, 1^{t(|x|)}) = x\}$$

where  $|\Pi|$  is referred to as the description length of  $\Pi$ .

It is well known that for every  $x$ ,  $K^t(x) \leq |x| + c$ , for some constant  $c$  depending only on the choice of the universal TM  $\mathcal{U}$ .

► **Fact 6.** For every universal TM  $\mathcal{U}$ , there exists a constant  $c$  such that for every  $x \in \{0, 1\}^*$ , and for every  $t$  such that  $t(n) > 0$ ,  $K_{\mathcal{U}}^t(x) \leq |x| + c$ .

We will use the following bound on the Kolmogorov complexity of strings sampled from the uniform distribution.

► **Lemma 7.** For any universal TM  $\mathcal{U}$  and every  $n \in \mathbb{N}$ , it holds that

$$\Pr_{x \leftarrow \{0, 1\}^n} [K_{\mathcal{U}}^t(x) \geq n - i] \geq 1 - 2^{-i}.$$

### 3 Definitions

Given some efficient threshold function  $s$ , let  $\text{MK}_{\mathcal{M}}^t\text{P}[s]$  denote the set of strings  $x$  s.t.  $K_{\mathcal{M}}^t(x) \leq s(|x|)$  (where we let  $K_{\mathcal{M}}^t(x) = \infty$  if there is no  $\Pi$  such that  $M(\Pi, 1^t) = x$ ). Let search- $\text{MK}_{\mathcal{M}}^t\text{P}[s]$  denote the search problem in which given a string  $x$  with  $K_{\mathcal{M}}^t(x) \leq s(|x|)$ , the output is a program  $\Pi$  of length at most  $s(|x|)$  with  $M(\Pi, 1^{t(n)}) = x$ .

We start with the definition of a black-box emulator and a black-box universal TM.

► **Definition 8** (Black-box emulator/Black-box Universal TM). A function  $M : \{0, 1\}^* \times 1^* \rightarrow \{0, 1\}^* \cup \{\perp\}$ , is a black-box TM emulator if  $M$  has “unique outputs”: For any  $\Pi \in \{0, 1\}^*$ ,  $t_1, t_2 \in \mathbb{N}$ ,  $t_1 \leq t_2$ , if  $M(\Pi, 1^{t_1}) \neq \perp$ ,  $M(\Pi, 1^{t_2}) = M(\Pi, 1^{t_1})$ . A black-box TM emulator  $\mathcal{U}$  is a black-box universal Turing machine (black-box UTM) if there exists a universal Turing machine  $\mathcal{U}_0$  such that for any  $(\Pi, 1^t)$ , if  $\Pi$  is a valid description of a Turing machine (w.r.t  $\mathcal{U}_0$ ), then  $\mathcal{U}(\Pi, 1^t) = \mathcal{U}_0(\Pi, 1^t)$ .

We next define black-box solvers for  $\text{MK}_{\mathcal{M}}^t\text{P}[s]$ . For a TM  $M$ , a function  $t : \mathbb{N} \rightarrow \mathbb{N}$ , and a number  $n \in \mathbb{N}$ , we let  $f_n^{M,t} : \{0, 1\}^{\leq 2n} \rightarrow \{0, 1\}^*$  be the function defined by  $f_n^{M,t}(\Pi) = M(\Pi, 1^{t(n)})$  for any  $\Pi \in \{0, 1\}^{\leq 2n}$ .

► **Definition 9** (Black-box  $\text{MK}^t\text{P}$ -solver). For functions  $t, s, T : \mathbb{N} \rightarrow \mathbb{N}$ , we say that  $\text{MK}^t\text{P}[s]$  admits a black-box  $\text{MK}^t\text{P}[s]$ -solver of size  $T(n)$  if for every black-box universal TM  $\mathcal{U}$ , there exists a circuit family  $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$  of size at most  $T(n)$ , such that for every  $n \in \mathbb{N}$ ,  $C_n$  is a  $f_n^{\mathcal{U},t}$ -oracle aided circuit that decides  $\text{MK}_{\mathcal{U}}^t\text{P}[s]$  on inputs of length  $n$ .

► **Remark 10** (Black-Box vs. Fully Black-Box solvers). In [14] there are two definitions for  $K^t$  solvers: fully black-box and (plain) black-box. The (plain) black-box is defined as in the definition above, while for fully black-box the circuit family  $\mathcal{C}$  can be used to compute the  $K_{\mathcal{U}}^t$  complexity with respect to any universal TM  $\mathcal{U}$ .

The construction of  $K^t$  solver given in [14] is not fully black-box. Indeed, the circuit family in [14] construct is dependent in the universal TM with respect to we defined  $K^t$ . Moreover, proving lower bounds for (plain) black-box solvers is stronger.

We define generalized black-box solver in exactly the same way except that we quantify over all black-box TM emulators (as opposed to just universal ones).

► **Definition 11** (Generalized black-box MK<sup>t</sup>P-solver). *For functions  $t, s, T: \mathbb{N} \rightarrow \mathbb{N}$ , we say that  $\text{MK}^t\text{P}[s]$  admits a generalized black-box MK<sup>t</sup>P[s]-solver of size  $T(n)$  if the following holds for every black-box TM  $M$ , there exists a circuit family  $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$  of size at most  $T(n)$ , such that for every  $n \in \mathbb{N}$ ,  $C_n$  is a  $f_n^{M,t}$ -oracle aided circuit that decides  $\text{MK}_M^t\text{P}[s]$  on inputs of length  $n$ .*

We similarly define black-box solvers and generalized black box solvers for search-MK<sup>t</sup>P.

## 4 Generalized Solvers Scaling with the Threshold

We show how to turn a black-box solver into a *generalized* black-box solver where the circuit size *scales with the threshold*. As mentioned before, proof follows standard techniques from the literature on hardness magnification (i.e., hashing down the statement to roughly the threshold size, and then applying the solver on the smaller instance and thereby improving the running time) [16, 2, 15].

► **Theorem 12.** *There exists  $q \in \text{poly}$  such that the following holds. Let  $T: \mathbb{N} \rightarrow \mathbb{N}$  be a function, and assume that for  $t(n) = n$  there exists a black-box  $\text{MK}^t\text{P}[n - 4]$  solver of size  $T(n)$ . Then, there exists a generalized black-box  $\text{MK}^t\text{P}[s]$  solver of size  $T(s(n) + 5) \cdot q(n)$  for every function  $s(\cdot)$  with  $s(n) \leq 2n$  and for every function  $t'(\cdot)$ .<sup>5</sup>*

**Proof of Theorem 12.** Fix an efficient universal TM  $U$ , and let  $p \in \text{poly}$  be such that  $p(n, t)$  bounds the size of a circuit implementing  $U(\Pi, 1^t)$  for inputs  $(\Pi, 1^t)$  with  $|\Pi| = n$ . Let  $M$ ,  $s(n)$ , and  $t'(n)$  be the TM, time function and threshold for which we want to solve  $\text{MK}_M^t\text{P}[s]$ . Let  $t(n) = n$ . For every  $n \in \mathbb{N}$ , let  $\mathcal{H}_n = \left\{ h: \{0, 1\}^n \rightarrow \{0, 1\}^{s(n)+5} \right\}$  be a pairwise independent hash family, such that there exists  $m \in \text{poly}$  for which  $m(n + s(n))$  bounds the circuit size evaluating  $h$  for every  $h \in \mathcal{H}_n$ . Fix  $n \in \mathbb{N}$ . We start by showing a distribution over circuits that solves  $\text{MK}_M^t\text{P}[s]$  with good probability.

For every  $h \in \mathcal{H}_n$ , we define  $U_h$  to be the following black-box universal TM:

$$U_h(\Pi, 1^t) = \begin{cases} U(\Pi_{>1}, 1^t) & \text{if } \Pi_1 = 0 \\ h(M(\Pi_{>1}, 1^{t'(n)})) & \text{if } |\Pi| \leq 2n + 1, \Pi_1 = 1 \text{ and } \left| M(\Pi_{>1}, 1^{t'(n)}) \right| = n \\ \perp & \text{Otherwise} \end{cases}$$

By the assumption that there exists a black-box solver of size  $T(n)$  for every black-box universal TM, there exists a circuit of size  $C_h^s$  of size  $T(s(n) + 5)$  that solves  $\text{MK}_{U_h}^t\text{P}[n - 4]$  on input of length  $n' = s(n) + 5$ , and using oracle to the function  $f_n^h: \{0, 1\}^{\leq 2n'} \rightarrow \{0, 1\}^*$  defined by  $f_n^h(\Pi) = U_h(\Pi, 1^{t'(n')})$ .

Let  $C_h$  be the circuit that given input  $x \in \{0, 1\}^n$ , computes  $h(x)$  and outputs  $C_h^s(h(x))$ . We claim that for every  $x \in \{0, 1\}^n$ ,

1. if  $K_M^{t'}(x) \leq s(n)$ ,  $C_h(x)$  outputs Yes for every  $h$ , and,
2. if  $K_M^{t'}(x) > s(n)$ , it holds that for  $h \leftarrow \mathcal{H}_n$ ,  $C_h(x)$  outputs No with probability at least  $3/4$ .

<sup>5</sup> We remark that the theorem extends also to the search versions of the same problems with essentially identically the same proof. Since we later will show a generic decision-to-search reduction, we omit the details.

To see (1), consider any  $x \in \{0, 1\}^n$  s.t.  $K_M^{t'}(x) \leq s(n)$ . Then there exists a program  $\Pi$  of length at most  $s(n)$  such that  $M(\Pi, 1^{t(n)}) = x$ . Therefore  $U_h(1|\Pi, 1^{t(n)}) = h(M(\Pi, 1^{t(n)})) = h(x)$  and thus  $K_{U_h}^t(h(x)) \leq s(n) + 1 \leq n' - 4$ , so  $C_h(x)$  will always answer Yes.

For (2), consider any  $x \in \{0, 1\}^n$  s.t.  $K_M^{t'}(x) > s(n)$ . We claim that with probability at least  $3/4$  over the choice of a random  $h \leftarrow \mathcal{H}_n$ , it holds that  $K_{U_h}^t(h(x)) > n' - 4$ , which implies that  $C_h(x)$  outputs No. To see the above, assume that for some  $h$ ,  $K_{U_h}^t(h(x)) \leq n' - 4$ . Then there exists  $\Pi$  such that  $|\Pi| \leq n' - 4$ , and  $U_h(\Pi, 1^{t(n)}) = h(x)$ . By the definition of  $U_h$ , it either holds that  $\Pi_1 = 0$ , and then  $K_{U_h}^t(h(x)) \leq n' - 5$ , or  $\Pi_1 = 1$ , which means that  $h(x) = h(x')$  for some  $x'$  with  $K_M^{t'}(x') \leq n' - 5 = s(n)$ . In the following we show that the probability that one of the above happens is at most  $1/4$  (over a random choice of  $h \leftarrow \mathcal{H}_n$ ). Indeed, since  $\mathcal{H}_n$  is a pairwise independent family,  $h(x)$  uniformly distributed when  $h \leftarrow \mathcal{H}_n$ . Therefore,

$$\Pr_{h \leftarrow \mathcal{H}_n} [K_{U_h}^t(h(x)) \leq n' - 4] = \Pr_{y \leftarrow \{0,1\}^{n'}} [K_U^t(y) \leq n' - 4] \leq 2^{-3}.$$

Moreover, for every  $x' \neq x$ , it holds that  $\Pr_{h \leftarrow \mathcal{H}_n} [h(x) = h(x')] \leq 2^{-s(n)-5}$ . By a union bound over all  $x'$  with  $K_M^{t'}(x') \leq s(n)$ , we get that the probability of collision  $h(x) = h(x')$  with such  $x'$  is at most  $2^{-4}$ . Using the union bound again, it holds that with probability at least  $1 - 2^{-3} - 2^{-4} > 3/4$ , both  $K_{U_h}^t(h(x)) > n' - 4$  and there is no  $x' \neq x$  with  $K_M^{t'}(x') \leq s(n)$  such that  $h(x) = h(x')$ . In this case,  $K_{U_h}^t(h(x)) > n' - 4$ , and  $C_h(h(x))$  answers No.

The proof now follows by simple amplification: for  $h_1, \dots, h_n \in \mathcal{H}_n$ , let  $C_{h_1, \dots, h_n}$  be the circuit that computes  $C_{h_1}(x), \dots, C_{h_n}(x)$  and outputs No if one of the execution output No. It follows using a standard Union bound, that with positive probability over the random choice of  $h_1, \dots, h_n \leftarrow \mathcal{H}_n$ ,  $C_{h_1, \dots, h_n}$  outputs the right answer for all  $x \in \{0, 1\}^n$ ; thus, there exists a fixed choice of  $h_1, \dots, h_n$  that works for every input.

We finally bound the size of  $C_{h_1, \dots, h_n}$ . We start with bounding the size of a circuit with  $f_n^h$  oracle, for every  $h \in \{h_1, \dots, h_n\}$ . In this case,

$$\begin{aligned} |C_{h_1, \dots, h_n}| &\leq n \cdot m(n + s(n)) + n \cdot |C_h^s| + O(n) \\ &\leq n \cdot m(n + s(n)) + n \cdot T(s(n) + 5) + O(n). \end{aligned}$$

Next, observe that each  $f_n^h$  oracle can be implemented using a circuit of size  $m(n + s(n)) + p(2n', 2n')$  using oracle to the function  $f_n^{M, t'}: \{0, 1\}^{\leq 2n} \rightarrow \{0, 1\}^*$  defined by  $f_n^{M, t'}(\Pi) = M(\Pi, 1^{t(n)})$ . Thus, the size of a  $f_n^{M, t'}$ -oracle aided circuit computing  $C_{h_1, \dots, h_n}$  is at most  $T(s(n) + 5) \cdot q(n)$  for  $q(n) = (m(n + s(n)))^2 + p(2(s(n) + 5), 2(s(n) + 5))$ . ◀

By taking  $T(n) = 2^{\alpha \cdot n \cdot \text{poly}(n)}$ , we get the following corollary.

► **Corollary 13.** *Assume the existence of a  $2^{\alpha n} \cdot \text{poly}(n)$ -size black-box solver for  $\text{MK}^t\text{P}[n-4]$  for  $t(n) = n$ . Then there exists a generalized black box  $\text{MK}^t\text{P}[s]$  solver of size  $2^{\alpha \cdot s(n)} \cdot \text{poly}(n)$  for all functions  $t'(\cdot)$  and  $s = s(n)$  with  $s(n) \leq 2n$ .*

## 5 From Decision to Search

In this section we show that if there exists a non-trivial black box solver to  $\text{MK}^t\text{P}$ , then such a solver (with roughly the same efficiency) exists also for search- $\text{MK}^t\text{P}$ .

► **Theorem 14.** *There exists  $q \in \text{poly}$  such that the following holds. Let  $p: \mathbb{N} \rightarrow \mathbb{N}$  be a monotone function, and let  $T: \mathbb{N} \rightarrow \mathbb{N}$  and  $t: \mathbb{N} \rightarrow \mathbb{N}$  be functions. Assume that for every  $s: \mathbb{N} \rightarrow \mathbb{N}$  with  $s(n) \leq 2n$  there exists a generalized black-box  $\text{MK}^t\text{P}[s]$  solver of size  $T(s(n)) \cdot p(n)$ . Then, there exists a generalized black-box search- $\text{MK}^t\text{P}[s]$  solver of size  $T(s(n) + \lceil \log n \rceil) \cdot p(n + s(n)) \cdot q(n)$  for every  $s: \mathbb{N} \rightarrow \mathbb{N}$  such that  $s(n) \leq 2n - \lceil \log n \rceil$ .*

**Proof.** Let  $M$  be a black-box TM emulator, and for every  $n \in \mathbb{N}$ , let  $M_n$  be the following black-box TM emulator. Given  $\Pi$  and  $1^{t'}$ ,  $M_n$  interprets  $\Pi = i|\Pi'$ , where the first  $\lceil \log n \rceil$  bits of  $\Pi$  interpreted as an index  $i \in [n]$ , and the rest of the bits interpreted as a program  $\Pi'$ . Then,  $M_n$  acts as follows:

$$M_n(i, \Pi', 1^{t'}) = \begin{cases} M(\Pi', 1^{t(n)})|\Pi'_{\leq i} & \text{if } i \leq n, |\Pi'| \leq 2n \text{ and } |M(\Pi', 1^{t(n)})| = n \\ \perp & \text{Otherwise} \end{cases}$$

We observe that for every  $x$ , for every  $i \in [n]$ , and for every program  $\Pi'$  of length  $\ell \leq 2n$  such that  $M(\Pi', 1^t) = x$ , it holds that  $K_M^t(x) + \lceil \log n \rceil \leq K_{M_n}^t(x|\Pi'_{\leq i}) \leq \ell + \lceil \log n \rceil$ . In particular, assuming that  $K_M^t(x) \leq 2n$ , for the minimal-length program  $\Pi'$  such that  $M(\Pi', 1^t) = x$  it holds that  $K_{M_n}^t(x|\Pi'_{\leq i}) = K_M^t(x) + \lceil \log n \rceil$ . Moreover, for every  $z \in \{0, 1\}^*$  such that  $z$  is not a prefix of a program of length at most  $\ell$  that outputs  $x$ , it holds that  $K_{M_n}^t(x|z) > \ell + \lceil \log n \rceil$ . We can thus use an algorithm that decides  $\text{MK}_{M_n}^t \text{P}$  to find a program  $\Pi$  of length at most  $s$  such that  $M(\Pi, 1^t) = x$ . This can be done by the following process:

1. Check if  $K_{M_n}^t(x) \leq s(n) + \lceil \log n \rceil$ . If not output  $\perp$ .
2. Let  $z = \perp$ .
3. For every  $i \in [s(n)]$ :
  - a. Check if  $M(z, 1^t) = x$ . If it does, output  $z$ .
  - b. Check if  $K_{M_n}^t(x|z|0) \leq s(n) + \lceil \log n \rceil$ , let  $z = z|0$ . Otherwise let  $z = z|1$ .
4. Output  $z$ .

Since  $K_{M_n}^t(x|z|0) \leq s(n) + \lceil \log n \rceil$  if and only if  $z|0$  is a prefix of a program  $\Pi$  of length at most  $s$  such that  $M(\Pi, 1^t) = x$ , the above process always finds such a program. We left to show that the above process can be implemented using a circuit of size  $T(s(n) + \lceil \log n \rceil) \cdot p(n + s(n)) \cdot \text{poly}(n)$ .

Let  $s'$  be the function defined by  $s'(k) = 1$  for every  $k < n$ , and  $s'(k) = s(n) + \lceil \log n \rceil$  otherwise. Then if  $s(n) \leq 2n - \lceil \log n \rceil$ , it holds that  $s'(k) \leq 2k$ . By our assumption, for every  $n'$  there exists a  $f_{n'}^{M_n, t}$ -oracle aided circuit  $C_{n'}$  of size  $T(s'(n')) \cdot p(n')$  that decides  $\text{MK}_{M_n}^t \text{P}[s']$  on inputs of length  $n'$ . We observe that the above process can be implemented with one call to each of  $C_{n'}$ , for  $n' \in \{n, \dots, n + s(n)\}$ . Moreover, the  $f_{n'}^{M_n, t}$ -oracle can be implemented by a poly-size circuit using an  $f_n^{M, t}$ -oracle. Thus, the above process can be implemented using a circuit of size at most  $s(n) \cdot T(s(n) + \lceil \log n \rceil) \cdot p(n + s(n)) \cdot \text{poly}(n)$ , as required.  $\blacktriangleleft$

By taking  $T(s(n)) = 2^{\alpha \cdot s(n)}$ , we get the following corollary.

**► Corollary 15.** *Assume there exists a generalized black box  $\text{MK}^t \text{P}[s]$  solver of size  $2^{\alpha \cdot s(n)} \cdot \text{poly}(n)$  for all functions  $t(\cdot)$  and  $s = s(n)$  with  $s(n) \leq 2n$ . Then there exists a  $2^{\alpha \cdot s(n)} \cdot \text{poly}(n)$ -size generalized black-box solver for search- $\text{MK}^t \text{P}[s]$  for every function  $t'(\cdot)$  and every function  $s(n) \leq 2n - \lceil \log n \rceil$ .*

---

## References

- 1 Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969.
- 2 Lijie Chen, Ce Jin, and R Ryan Williams. Hardness magnification for all sparse np languages. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1240–1255. IEEE, 2019.
- 3 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal on Computing*, 29(3):790–803, 2000.

- 4 Gudmund Skovbjerg Frandsen and Peter Bro Miltersen. Reviewing bounds on the circuit size of the hardest functions. *Information processing letters*, 95(2):354–357, 2005.
- 5 J. Hartmanis. Generalized kolmogorov complexity and the structure of feasible computations. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 439–445, November 1983. doi:10.1109/SFCS.1983.21.
- 6 Shuichi Hirahara, Rahul Ilango, and Ryan Williams. Beating brute force for compression problems. Technical Report TR23-171, Electronic Colloquium on Computational Complexity, 2023.
- 7 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for np. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 104–114. IEEE, 2011.
- 8 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79, 2000.
- 9 Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. doi:10.1016/0304-3975(86)90081-2.
- 10 A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.
- 11 Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1243–1254. IEEE, 2020.
- 12 Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded kolmogorov complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 722–735, 2021.
- 13 Oleg B Lupanov. On a method of circuit synthesis. *Izvestia VUZ*, 1:120–140, 1958.
- 14 Noam Mazon and Rafael Pass. The non-uniform peregbor conjecture for time-bounded kolmogorov complexity is false. *15th Innovations in Theoretical Computer Science*, 2024.
- 15 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Theory OF Computing*, 17(CCC 2019 Special Issue), 2021.
- 16 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 65–76. IEEE, 2018.
- 17 Hanlin Ren and Rahul Santhanam. Hardness of kt characterizes parallel cryptography. In *36th Computational Complexity Conference (CCC 2021)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.
- 18 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.
- 19 R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964. doi:10.1016/S0019-9958(64)90223-2.
- 20 Boris A Trakhtenbrot. A survey of russian approaches to peregbor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.
- 21 AC-C Yao. Coherent functions and program checkers. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 84–94, 1990.

## **A** search- $MK^tP$ and Function Inversion

We observe that generalized black-box solvers for search- $MK^tP$  directly yield function inverters with roughly the same complexity, and vice versa.

► **Lemma 16.** *There exists  $p \in \text{poly}$  such that the following holds. Assume that for some  $t = t(n)$  there exists a generalized black-box search- $MK^tP[n]$  solver of size  $T = T(n)$ . Then for every function  $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$  there exists a  $\pi$ -oracle aided circuit of size  $T(n) \cdot p(n)$  that inverts  $\pi$ .*

**Proof.** Let  $M$  be the black box TM that on input  $x \in \{0, 1\}^n$ ,  $1^t$  outputs  $y = \pi(x)$ . By assumption, there exists a  $f_n^{M,t}$ -oracle aided circuit of size at most  $T(n)$ , that given an input  $y \in \{0, 1\}^n$  finds an input  $x$  of length at most  $n$ , such that  $M(x, 1^{t(n)}) = y$ , if such exists. Since  $M(x, 1^{t(n)}) = \pi(x)$ , such an  $x$  is a pre-image of  $y$ . Moreover, by the definition of  $M$ , the  $f_n^{M,t}$ -oracle can be implemented efficiently using a  $\pi$ -oracle. ◀

The converse of Lemma 16 was implicitly proven in [14]; we repeat the proof for the convenience of the reader.

► **Lemma 17.** *There exists  $p \in \text{poly}$  such that the following holds. Assume that for every function  $\pi: \{0, 1\}^n \rightarrow \{0, 1\}^n$  there exists a  $\pi$ -oracle aided circuit of size  $T(n)$  that inverts  $\pi$  with probability 1 (for every  $y = \pi(x)$ ,  $f(C(y)) = y$ ). Then there exists a black-box  $\text{MK}^t\text{P}[s]$  solver of size  $T(n + \lceil \log n \rceil) \cdot p(n)$  for every  $t: \mathbb{N} \rightarrow \mathbb{N}$  and every  $s: \mathbb{N} \rightarrow \mathbb{N}$  with  $s(n) \leq n$ .*

**Proof.** Let  $U$  be a black-box universal TM. Let  $f'_n: \{0, 1\}^n \times [n] \rightarrow \{0, 1\}^n \times [n]$  be defined as

$$f'_n(\Pi, i) = \begin{cases} (U(\Pi_{\leq i}, 1^{t(n)}), i) & |U(\Pi_{\leq i}, 1^{t(n)})| = n \\ 0^n & \text{Otherwise} \end{cases}$$

Let  $n' = n + \lceil \log n \rceil$ . In the following, we assume that both the domain and the range of  $f'_n$  is  $\{0, 1\}^{n'}$ , by the use of appropriate encoding and padding. By assumption, there is a circuit family  $\widehat{C} = \{\widehat{C}_n\}_{n \in \mathbb{N}}$  with  $f'_n$  oracle, of size  $T(n')$  that inverts  $f'_n$  with probability 1.

Given a circuit  $\widehat{C}_n$  that inverts  $f'_n$ , we can construct a ( $f'_n$ -oracle aided) circuit  $C_n$  that computes the  $K_{\cup}^t$  complexity of any string  $x$  of length  $n$  with  $K_{\cup}^t(x) \leq n$ . This can be done by computing  $f'^{-1}_n(x, 1), \dots, f'^{-1}_n(x, n)$  and outputting Yes if there exists  $(\Pi, i)$  such that  $U(\Pi_{\leq i}, 1^{t(n)}) = x$  and  $i \leq s(n)$  (the  $t$ -bounded Kolmogorov complexity of the string  $0^n$  can be hardcoded in the circuit).

Observe that the size of  $C_n$  is  $n' \cdot |\widehat{C}_n| + \text{poly}(n)$ . Thus, there exists a circuit family of size  $n' \cdot T(n') + \text{poly}(n) = T(n') \cdot \text{poly}(n)$ , with  $f'_n$  oracle, that solves  $\text{MK}^t_{\cup}\text{P}[s]$ . Lastly, observe that  $f'_n$  can be efficiently computed from  $f_n^{U,t}$ , thus we can replace the  $f'_n$  oracle with a small circuit using an  $f_n^{U,t}$ -oracle, to get a circuit of size  $T(n + \lceil \log n \rceil)\text{poly}(n)$ . ◀

## B MCSP[s] as a special case of $\text{MK}^t_{\text{M}}\text{P}$

We note that any generalized black-box  $\text{MK}^t\text{P}[s]$  solver can be used to solve  $\text{MCSP}[s]$ . In fact, we observe that the  $\text{MCSP}[s]$  problem can be formulated as a  $\text{MK}^t_{\text{M}}\text{P}[s']$  instance for a particular choice of an (efficient but non-universal) TM  $M$ , and for a function  $s'(n) \approx s(n)$ .

Towards this, we will rely on the fact that circuits can be *succinctly* encoded as bit strings from which the circuit can be efficiently decoded. In particular, as observed in [17, 6], the encoding from [4] satisfies this requirement.

► **Lemma 18** (Implicit in [4], see also [17, 6]). *There exists an efficiently computable function  $\ell(s, k) \in (1 + o(1))(s \cdot \log(s + k))$  such that  $\ell$  is monotone in  $s$  and the following holds. There exists an efficient algorithm  $\text{Dec}$ , such that for every circuit  $C: \{0, 1\}^k \rightarrow \{0, 1\}$  of size  $s$ , there exists  $x \in \{0, 1\}^{\ell(s,k)}$  such that  $\text{Dec}(x)$  is a circuit of size  $s$  that computes the same function as  $C$ . Moreover, for every  $x$  such that  $\text{Dec}(x)$  outputs a circuit  $C: \{0, 1\}^k \rightarrow \{0, 1\}$  of size  $s$ , it holds that  $|x| = \ell(s, k)$ .<sup>6</sup>*

<sup>6</sup> Note that we here requires the length of an encoding of a circuit of size  $s$  to be *exactly*  $\ell(s, k)$  (in contrast to bounded by  $\ell(s, k)$ ). As far as we can tell, this property has not been previously stated but it can be assumed without loss of generality using padding, and by assuming that given an input  $x$ ,  $\text{Dec}$  only outputs a circuit  $C$  of size  $s$  if it holds that  $|x| = \ell(s, k)$ , or outputs  $\perp$  otherwise.

## 66:12 On Black-Box Meta Complexity and Function Inversion

We now observe that the MCSP is a special-case of the  $\text{MK}_M^t\text{P}$  problem for a specific choice of the TM  $M$ .

► **Lemma 19.** *There exists an efficient TM  $M$  such that the following holds for every  $s: \mathbb{N} \rightarrow \mathbb{N}$  and every  $t: \mathbb{N} \rightarrow \mathbb{N}$  with  $t(n) \geq n$ . Deciding  $\text{MCSP}[s]$  is equivalent to deciding  $\text{MK}_M^t\text{P}[s']$ , for  $s'(n) = \ell(s(n), \lfloor \log n \rfloor)$ .*

**Proof.** Let  $Dec$  be the function from Lemma 18, and let  $M$  be the TM that given an input  $x, 1^t$ , computes  $Dec(x)$  to get a circuit  $C: \{0, 1\}^k \rightarrow \{0, 1\}$ . If  $x$  is not valid encoding of a circuit, or  $2^k \neq |x|$ ,  $M$  outputs  $\perp$ . If  $t \leq 2^k$ ,  $M$  also outputs  $\perp$ . Otherwise,  $M$  outputs the truth table of  $C$ . Since  $\ell$  is monotone in  $s$ , there exists a program of length less than  $s'(n)$  if and only if there exists a circuit of size less than  $s(n)$  for  $x$ . ◀



# Explicit and Near-Optimal Construction of $t$ -Rankwise Independent Permutations

Nicholas Harvey  

Department of Computer Science and Department of Mathematics,  
University of British Columbia, Vancouver, Canada

Arvin Sahami  

Department of Computer Science and Department of Mathematics,  
University of British Columbia, Vancouver, Canada

---

## Abstract

Letting  $t \leq n$ , a family of permutations of  $[n] = \{1, 2, \dots, n\}$  is called  $t$ -rankwise independent if for any  $t$  distinct entries in  $[n]$ , when a permutation  $\pi$  is sampled uniformly at random from the family, the order of the  $t$  entries in  $\pi$  is uniform among the  $t!$  possibilities.

Itoh et al. show a lower bound of  $(n/2)^{\lfloor \frac{t}{4} \rfloor}$  for the number of members in such a family, and provide a construction of a  $t$ -rankwise independent permutation family of size  $n^{O(t^2/\ln(t))}$ .

We provide an explicit, deterministic construction of a  $t$ -rankwise independent family of size  $n^{O(t)}$  for arbitrary parameters  $t \leq n$ . Our main ingredient is a way to make the elements of a  $t$ -independent family “more injective”, which might be of independent interest.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Pseudorandomness and derandomization

**Keywords and phrases** Rankwise independent permutations

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.67

**Category** RANDOM

## 1 Introduction

An important topic in the area of pseudorandomness is the construction of random variables such that any  $t$  of them are independent (for some parameter  $t \in \mathbb{N}$ ), given a small source of purely random bits. A fundamental notion introduced by Wegman and Carter in 1979 [2] is that of a  $t$ -independent family<sup>1</sup>, defined as follows (see also [9, Definition 3.31]).

► **Definition 1** ( $t$ -independent family). *Let  $m, n, t$  be positive integers with  $t \leq m$ . A family  $\mathcal{H}$  of functions mapping  $[m] \rightarrow [n]$  is called  $t$ -independent if, when  $h \in \mathcal{H}$  is chosen uniformly at random, for any  $t$  distinct  $x_1, \dots, x_t \in [m]$  and  $t$  elements  $y_1, \dots, y_t \in [n]$ ,*

$$\mathbb{P}(h(x_i) = y_i \text{ for } i = 1, \dots, t) = \frac{1}{n^t},$$

*or equivalently, that the  $t$  random variables  $h(x_1), \dots, h(x_t)$  are independently and uniformly distributed in  $[n]$ .*

These  $t$ -independent families are well-studied, and have found various applications. One example is to derandomize a randomized algorithm that uses certain independent random variables, but one can relax the assumption of being *mutually* independent to any  $t$  of them being independent. Then often one can derandomize the algorithm by iterating over the elements of  $\mathcal{H}$  to find a function for which the algorithm succeeds. See [9, section 3.5] for such

---

<sup>1</sup> Throughout this paper, we use the term “family” to refer to a multiset, meaning that the members need not be distinct.



© Nicholas Harvey and Arvin Sahami;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 67; pp. 67:1–67:12



Leibniz International Proceedings in Informatics

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

an application for the MaxCut problem. It is then desirable to have explicit constructions of such a family  $\mathcal{H}$  with small size. In fact, explicit constructions of such families of near-optimal size are known [4] for any parameters  $1 \leq t \leq m$  and any  $n$ .

One can also define analogous families when restricting to permutations of  $[n]$  instead of general functions. This natural restriction yields the notion of  $t$ -independent permutations.

► **Definition 2** ( *$t$ -independent permutation*). *A family  $\Pi$  is called  $t$ -independent if it contains permutations of  $[n]$  such that, for any  $t$  distinct  $x_1, \dots, x_t \in [n]$  and any  $t$  distinct elements  $y_1, \dots, y_t \in [n]$ ,*

$$\mathbb{P}(\pi(x_i) = y_i \text{ for } i = 1, \dots, t) = \prod_{i=0}^{t-1} \frac{1}{n-i}$$

when  $\pi \in \Pi$  is chosen uniformly at random.

Explicit construction of such families with a small size, namely such that  $|\Pi| \leq n^{O(t)}$ , remains an open problem. This bound is near-optimal, since there is an obvious lower bound of  $|\Pi| \geq \prod_{i=0}^{t-1} (n-i)$ , which follows from the definition.

In fact, there are few non-trivial constructions of such families for any  $t \geq 4$ . Perhaps the closest result in this direction is a probabilistic proof for the existence of small (i.e., with  $|\Pi| \leq n^{O(t)}$ )  $t$ -independent permutations for any  $1 \leq t \leq n$  due to Kuperberg, Lovett and Peled [7]. However, their proof does not seem to yield an efficient deterministic or randomized construction of the family, as it has a tiny success probability.

Many relaxed notions related to  $t$ -independence have been proposed for permutation families, including “ $t$ -restricted min-wise independent” [1] and “ $t$ -rankwise independent” families [5]. The latter is the focus of this paper.

► **Definition 3** ( *$t$ -rankwise independent permutation*). *A family  $\Pi$  of permutations over  $[n]$  is called  $t$ -rankwise independent if for any  $t$  distinct points  $x_1, \dots, x_t \in [n]$ ,*

$$\mathbb{P}(\pi(x_1) < \pi(x_2) < \dots < \pi(x_t)) = \frac{1}{t!}$$

when  $\pi \in \Pi$  is chosen uniformly at random.

Another interesting type of permutation families has recently been proposed in the cryptography community. This is the notion of a *perfect sequence covering array* (PSCA).

► **Definition 4** (Yuster [10]). *Let  $t \leq n$ . The family  $\Pi$  of permutations of  $[n]$  is called a PSCA( $n, t$ ) if there exists a fixed  $\lambda \in \mathbb{N}$  such that for any  $t$  distinct indices  $i_1, \dots, i_t \in [n]$ , there are exactly  $\lambda$  permutations  $\pi \in \Pi$  such that*

$$(i_1, i_2, \dots, i_t) \text{ is a subsequence of } (\pi(1), \pi(2), \dots, \pi(n)).$$

(The notation and wording have been adapted to match ours.)

Let  $g^*(n, t)$  denote the smallest size of a PSCA family  $\Pi$ . Naturally, researchers in this field are interested in the value of  $g^*(n, t)$ , and in the construction of families that asymptotically achieve this minimum size.

It was observed in [6] that  $t$ -rankwise independent families and PSCAs are isomorphic. Specifically,  $\Pi$  is a PSCA( $n, t$ ) family if and only if  $\Pi^{-1} = \{\pi^{-1} : \pi \in \Pi\}$  is a  $t$ -rankwise independent family of permutations over  $[n]$ . Consequently, our construction of  $t$ -rankwise independent permutations can immediately be translated into a construction of PSCAs. Henceforth we will only use the terminology of  $t$ -rankwise independent families, and will no longer refer to PSCAs.

Itoh et al. [5] show a lower bound of  $(n/2)^{\lfloor \frac{t}{4} \rfloor} \leq |\Pi|$  for the size of a  $t$ -rankwise independent family  $\Pi$ . They also construct a family  $\Pi$  with  $|\Pi| \leq n^{O(t^2/\ln(t))}$ , which does not asymptotically match the lower bound.

We present a deterministic algorithm for constructing a  $t$ -rankwise independent family  $\Pi$  of permutations over  $[n]$ , with  $|\Pi| \leq n^{O(t)}$ . This asymptotically matches the known lower bound. Formally, the following is our main result.

► **Theorem 5 (Main).** *There exists a constant  $C > 0$  such that the following is true. Let  $n, t$  be positive integers with  $t \leq n$ . Then there exists a  $t$ -rankwise independent family  $\Pi$  consisting of permutations of  $[n]$  such that  $|\Pi| \leq (Cn)^{35t}$ . Furthermore, the whole family can be constructed by a deterministic algorithm in  $n^{O(t)}$  time. (The implied constant in the  $O(\cdot)$  notation does not depend on either  $n$  or  $t$ ).*

Our construction starts in Section 2.2 with a  $t$ -independent family  $\mathcal{H}$ , based on Reed-Solomon codes. The next step, appearing in Section 2.3, modifies it to obtain another  $t$ -independent family  $\mathcal{G}$  whose members, roughly speaking, look “more injective”. This step is the main technical contribution of the paper, and might be of independent interest. (Note that, since  $\mathcal{G}$  is a  $t$ -independent family, not all the maps in  $\mathcal{G}$  can be injective). Finally, in Section 2.4, we use this  $t$ -independent family  $\mathcal{G}$  to construct permutations of  $[n]$ , yielding the  $t$ -rankwise independent family  $\Pi$ .

## 2 The construction

### 2.1 Overview

Our construction involves three steps, which build upon each other.

1. Construct  $\mathcal{H}$ , a  $t$ -independent family of  $[n] \rightarrow \mathbb{Z}_N$  maps, where  $N = \Theta(n^3)$ .
2. Construct  $\mathcal{G}$ , a  $t$ -independent family of  $[n] \rightarrow \mathbb{Z}_N$  maps, such that each map’s image has size at least  $n - 16t$ . Intuitively, this condition says that each map has very few collisions, or is almost injective. (Being injective is equivalent to the image having size exactly  $n$ ).
3. Construct  $\Pi$ , a  $t$ -rankwise independent family of permutations on  $[n]$ .

The most substantial of these steps is the construction of  $\mathcal{G}$ , whereas the construction of  $\mathcal{H}$  is the most trivial. We explain these steps in the following sections.

### 2.2 Construction of $\mathcal{H}$

The construction of  $\mathcal{H}$  is standard. The first step is to find a prime  $p$  in the interval  $[n^3, 2n^3]$ . This must exist, by Bertrand’s postulate, and can be found in  $\tilde{O}(n^3)$  time using exhaustive search and a deterministic primality test. We set  $N = p$ , and therefore

$$n^3 \leq N \leq 2n^3. \tag{1}$$

Let  $\mathcal{H}$  be the family of  $[n] \rightarrow \mathbb{F}_N$  maps defined by polynomials over  $\mathbb{F}_N$  of degree less than  $t$ , namely

$$\mathcal{H} = \left\{ \sum_{0 \leq i \leq t-1} a_i x^i : a_i \in \mathbb{F}_N \right\}.$$

This family is well-known to be  $t$ -independent; see, e.g., [3, Exercise 5.8]. Note that the size of the family is  $|\mathcal{H}| = p^t = N^t$ .

### 2.3 Construction of $\mathcal{G}$

The next step is to use the family  $\mathcal{H}$  to build a family  $\mathcal{G}$ . Each map in  $\mathcal{H}$  will yield exactly one map in  $\mathcal{G}$ . The family  $\mathcal{G}$  will retain  $\mathcal{H}$ 's property of being  $t$ -independent. In addition, we will be able to guarantee that every map in  $\mathcal{G}$  has image size at least  $n - 16t$ . Thus each map has few collisions (although this is an informal term that we have not yet defined).

The family  $\mathcal{G}$  has a simple form, and it is constructed by the pseudocode shown in Algorithm 1. This algorithm computes a single, specific map  $\alpha : [n] \rightarrow \mathbb{Z}_N$ , then it constructs

$$\mathcal{G} = \{ h + \alpha : h \in \mathcal{H} \}.$$

▷ **Claim 6.** For any map  $\alpha$ , the resulting family  $\mathcal{G}$  will be  $t$ -independent.

*Proof.* Suppose that  $h$  is chosen uniformly at random from  $\mathcal{H}$ . For any  $t$  distinct entries  $x_1, \dots, x_t \in [n]$ ,  $\{h(x_i)\}_{i \in [t]}$  are independent, and hence  $\{f_i(h(x_i))\}_{i \in [t]}$  are independent for any deterministic functions  $f_i$ . In particular, since  $\alpha$  is not random, letting  $f_i(z) = z + \alpha(x_i)$ , we have that  $\{h(x_i) + \alpha(x_i)\}_{i \in [t]}$  remain independent. Lastly, for any  $k \in [n]$ ,  $h(k) + \alpha(k)$  is uniformly distributed since  $h(k)$  is uniform in  $\mathbb{Z}_N$ , and  $\alpha$  is not random. Thus  $\{(h + \alpha)(x_i)\}_{i \in [t]}$  are independent and uniform in  $\mathbb{Z}_N$ , as desired. ◁

We will prove that there is a specific choice of  $\alpha$  such that every  $h \in \mathcal{H}$  satisfies

$$|(h + \alpha)([n])| = |\{ h(x) + \alpha(x) : x \in [n] \}| \geq n - 16t,$$

which is the desired property of the family  $\mathcal{G}$ . In fact, it is possible to show that a random choice of  $\alpha$  will satisfy this property with positive probability. However, this would not quite achieve the goals of this paper, since ultimately we want an explicit, deterministic construction of a  $t$ -rankwise independent family of permutations. Instead, we will obtain a deterministic construction by derandomizing the randomized construction of  $\alpha$ .

Algorithm 1 contains pseudocode for this procedure, which we now briefly explain. The algorithm computes the values  $\alpha(1), \alpha(2), \dots, \alpha(n)$  one-by-one, in that order. Thinking of  $h + \alpha$  as mapping the “balls”  $[n]$  to the “bins”  $\mathbb{Z}_N$ , then  $S_k^h$  is the set of bins that have already received balls (for this particular function  $h$ ). In order to be as injective as possible, we want to avoid a collision (for every  $h$ ) between the  $k^{\text{th}}$  ball and these bins – that is, we want  $(h + \alpha)(k) \notin S_k^h \forall h \in \mathcal{H}$ . To do so, the algorithm uses a potential function (shown in (2)) in which the variable  $x$  corresponds to the value that will be used for  $\alpha(k)$ . This function penalizes any value  $x$  which would cause any further collision among any function  $h \in \mathcal{H}$ . This potential function is essentially a pessimistic estimator, as explained in Section 2.3.1 below.

► **Lemma 7.** *Algorithm 1 returns a  $t$ -independent family  $\mathcal{G}$  satisfying the following.*

$$|g([n])| \geq n - 16t \quad \forall g \in \mathcal{G}$$

The subset of the codomain that experienced a “collision” is defined to be

$$\mathcal{Y} = \{ y \in \mathbb{Z}_N : |g^{-1}(y)| \geq 2 \},$$

and the subset of the domain involved in these collisions is defined to be

$$\mathcal{X} = \bigcup_{y \in \mathcal{Y}} g^{-1}(y) = g^{-1}(\mathcal{Y}).$$

► **Corollary 8.** *The family  $\mathcal{G}$  produced by Lemma 7 satisfies  $|\mathcal{X}| \leq 32t$ .*

■ **Algorithm 1** Main Algorithm.

---

**Input:**  $t$ -independent family  $\mathcal{H}$  of  $[n] \rightarrow \mathbb{Z}_N$  maps s.t.  $|\mathcal{H}| = N^t$ .  
**Output:**  $t$ -independent family  $\mathcal{G}$  of  $[n] \rightarrow \mathbb{Z}_N$  maps s.t.  $|\mathcal{G}| = N^t$ ,  $|g([n])| \geq n - 16t \forall g \in \mathcal{G}$ .

- 1:  $\lambda \leftarrow \ln(16tN/n^2)$
- 2:  $\mathcal{G} \leftarrow \emptyset$
- 3: **for**  $k = 1, \dots, n$  **do**
- 4:     ▷ Compute the value  $\alpha(k)$
- 5:     **for**  $h \in \mathcal{H}$  **do**
- 6:         Let  $S_k^h = \{h(i) + \alpha(i) : 1 \leq i \leq k-1\} \subseteq \mathbb{Z}_N$ , and note that  $S_1^h = \emptyset$ .  
        This is  $(h + \alpha)([k-1])$ , the set of values that already appear in the image of  $h + \alpha$ .
- 7:         Define
 
$$\beta_k^h(\alpha(1), \alpha(2), \dots, \alpha(k-1), x) = \begin{cases} 1 & \text{if } h(k) + x \in S_k^h \\ 0 & \text{otherwise} \end{cases}$$

To ease notation, we will use the shorthand  
 $\beta_k^h(x) = \beta_k^h(\alpha(1), \alpha(2), \dots, \alpha(k-1), x)$ .
- 9:     **end for**
- 10:     Pick
 
$$a \in \operatorname{argmin}_{x \in \mathbb{Z}_N} \sum_{h \in \mathcal{H}} \exp\left(\lambda\left(\beta_k^h(x) + \sum_{1 \leq i \leq k-1} \beta_i^h(\alpha(1), \dots, \alpha(i))\right)\right) \quad (2)$$
- 11:     Let  $\alpha(k) \leftarrow a$
- 12: **end for**
- 13: **return** the family  $\mathcal{G} = \{h + \alpha : h \in \mathcal{H}\}$ .

---

A formal proof is in Appendix A, and here we present only a sketch.

**Proof (Sketch).** The size of  $\mathcal{X}$  is maximized by having exactly  $16t$  bins containing exactly 2 balls, and  $n - 32t$  bins containing exactly 1 ball. ◀

### 2.3.1 Proof of Lemma 7

For each function  $h \in \mathcal{H}$  and integer  $k \in [n]$ , there is a function  $\beta_k^h: \mathbb{Z}_N^k \rightarrow \{0, 1\}$  that is defined in Algorithm 1, and which we define equivalently here as

$$\beta_k^h(x_1, \dots, x_k) = \begin{cases} 1 & \text{if } \exists 1 \leq i \leq k-1 \text{ s.t. } h(k) + x_k = h(i) + x_i \pmod{N} \\ 0 & \text{otherwise.} \end{cases}$$

We will use the notation  $\beta_k^h(x_k)$  for  $\beta_k^h(x_1, \dots, x_k)$  when  $x_1, \dots, x_{k-1}$  are clear from context.

The scalar  $\lambda > 0$  is as defined as in Algorithm 1. Additionally, define the scalar  $c_\lambda > 0$  and the function  $\psi_k: \mathbb{Z}_N^k \rightarrow \mathbb{R}^+$  by

$$c_\lambda = \mathbb{E} \exp(\lambda Y) > 0$$

$$\psi_k(x_1, \dots, x_k) = \sum_{h \in \mathcal{H}} \exp\left(\lambda \sum_{i=1}^k \beta_i^h(x_1, \dots, x_i)\right) \cdot c_\lambda^{n-k}, \quad (3)$$

where  $Y$  is a random variable having the Bernoulli distribution with parameter  $n/N$ , which we write as  $\text{Bern}(n/N)$ . We will often write  $\psi_k(x_k)$  instead of  $\psi_k(x_1, \dots, x_k)$  for notational convenience.

Intuitively,  $\psi_k(x_1, \dots, x_k)$  is a pessimistic estimator of the expected number of functions  $h \in \mathcal{H}$  which would have  $|(h + \alpha)([n])| > n - 16t$  given that  $\alpha(i) = x_i \forall i \in [k]$ , and that the rest of the entries  $\alpha(k+1), \dots, \alpha(n)$  are chosen uniformly at random from  $\mathbb{Z}_N$ .

Let  $\alpha: [n] \rightarrow \mathbb{Z}_N$  be the mapping constructed by Algorithm 1.

▷ **Claim 9.**  $\psi_0 \geq \psi_1(\alpha(1)) \geq \psi_2(\alpha(2)) \geq \dots \geq \psi_n(\alpha(n))$ , where here we use the notation  $\psi_i(\alpha(i))$  to denote  $\psi_i(\alpha(1), \alpha(2), \dots, \alpha(i))$ .

▷ **Claim 10.**  $1 > \psi_0 = \exp(-16\lambda t) \cdot |\mathcal{H}| \cdot [\mathbb{E} \exp(\lambda Y)]^n$ .

Together, Claims 9 and 10 imply that

$$1 > \psi_n(\alpha(n)) = \sum_{h \in \mathcal{H}} \exp\left(\lambda \left(\sum_{i=1}^k \beta_i^h(\alpha(i))\right) - 16\lambda t\right).$$

Since all summands are non-negative, it follows that, for every  $h \in \mathcal{H}$ , we have

$$\exp\left(\lambda \left(\sum_{i=1}^k \beta_i^h(\alpha(i))\right) - 16\lambda t\right) < 1.$$

Observe that  $\sum_{i \leq k} \beta_i^h(\alpha(i)) = k - |S_k^h| \forall k, h$ . Taking the log and rearranging, we obtain that

$$n - |S_n^h| = \sum_{i=1}^n \beta_i^h(\alpha(i)) < 16t \quad \forall h \in \mathcal{H}.$$

Let  $g = h + \alpha$ . Since  $|g([n])| = |S_n^h|$ , we have  $|g([n])| > n - 16t$  for all  $h \in \mathcal{H}$ . This completes the proof of Lemma 7.

**Proof of Claim 9.** We will show that  $\psi_k(\alpha(k)) \leq \psi_k(\alpha(k-1)) \forall 1 \leq k \leq n$ . So let  $k \in [n]$  be arbitrary.

Our first observation is that, in the algorithm's iteration  $k$ , it chooses the value  $a = \alpha(k)$  to minimize  $\psi_k(\alpha(1), \dots, \alpha(k-1), a)$ . This holds because the functions

$$\sum_{h \in \mathcal{H}} \exp\left(\lambda \beta_k^h(x) + \lambda \sum_{i=1}^{k-1} \beta_i^h(\alpha(i))\right) \quad \text{and} \quad \psi_k(\alpha(1), \alpha(2), \dots, \alpha(k-1), x)$$

are positive multiples of each other.

Since  $\alpha(k)$  minimizes  $\psi_k$ , we clearly have

$$\psi_k(\alpha(1), \dots, \alpha(k)) \leq \mathbb{E}_{U \sim \text{Unif}(\mathbb{Z}_N)} \psi_k(\alpha(1), \dots, \alpha(k-1), U),$$

where  $\text{Unif}(S)$  denotes the uniform distribution on the set  $S$ . Hence in order to show that  $\psi_k(\alpha(k)) \leq \psi_{k-1}(\alpha(k-1))$ , it suffices to prove that

$$\mathbb{E}_{U \sim \text{Unif}(\mathbb{Z}_N)} \psi_k(\alpha(1), \dots, \alpha(k-1), U) \leq \psi_{k-1}(\alpha(k-1)). \quad (4)$$

Since  $\psi_k$  and  $\psi_{k-1}$  are both sums over  $h \in \mathcal{H}$ , it will suffice to prove this inequality for each summand. More specifically, we will ignore the  $e^{-16\lambda t}$  constant and define

$$\psi_k^h(x) = \exp\left(\lambda \sum_{i=1}^{k-1} \beta_i^h(\alpha(i)) + \lambda \beta_k^h(x)\right) \cdot c_\lambda^{n-k},$$

where, as above,  $c_\lambda = \mathbb{E} \exp(\lambda Y)$ , and  $Y$  is  $\text{Bern}(n/N)$ . Towards our inductive proof, we may rewrite this as

$$\psi_k^h(x) = \psi_{k-1}^h(\alpha(k-1)) \cdot \frac{1}{c} \cdot \exp(\lambda \beta_k^h(\alpha(1), \dots, \alpha(k-1), x)).$$

Plugging this into our goal (4), it suffices to prove that

$$\mathbb{E}_{U \sim \text{Unif}(\mathbb{Z}_N)} \psi_{k-1}^h(\alpha(k-1)) \cdot \frac{1}{c} \cdot \exp(\lambda \beta_k^h(\alpha(1), \dots, \alpha(k-1), U)) \leq \psi_{k-1}(\alpha(k-1)),$$

or equivalently (observing that  $\psi_{k-1}^h(\alpha(k-1)) > 0$ ),

$$\mathbb{E}_{U \sim \text{Unif}(\mathbb{Z}_N)} \exp(\lambda \beta_k^h(\alpha(1), \dots, \alpha(k-1), U)) \leq c_\lambda = \mathbb{E} \exp(\lambda Y). \quad (5)$$

Note that there are exactly  $|S_k^h|$  values of  $U$  that result in  $\beta_k^h(\alpha(1), \alpha(2), \dots, \alpha(k-1), U)$  taking the value 1, whereas the rest result in the value 0. Since  $U$  is uniformly distributed on  $\mathbb{Z}_N$  and  $|S_k^h| \leq n$  for all  $k \in [n]$ ,  $h \in \mathcal{H}$ , it follows that  $\beta_k^h(\alpha(1), \dots, \alpha(k-1), U)$  has a Bernoulli distribution  $\text{Bern}(p)$  where  $p \leq n/N$ . Since  $Y$  has the distribution  $\text{Bern}(n/N)$ , the desired inequality (5) follows.  $\triangleleft$

For the next proof, we will require the following statement of the Chernoff bound. A proof is given in Appendix A.

► **Theorem 11** (Poisson tail of Chernoff bound). *Let  $Y_1, \dots, Y_n$  be independent random variables supported on  $[0, 1]$ . Let  $\mu = \mathbb{E} \sum_{i=1}^n Y_i$ . Then, for any  $\delta \geq 1$ , if  $\lambda = \ln(1 + \delta)$  then*

$$\mathbb{P} \left( \sum_{i=1}^n Y_i \geq (1 + \delta)\mu \right) \leq \mathbb{E} \exp \left( \lambda \sum_{i=1}^n Y_i - \lambda(1 + \delta)\mu \right) \leq (1 + \delta)^{-(1+\delta)\mu/4}.$$

Proof of Claim 10. Let  $Y_1, \dots, Y_n$  be i.i.d.  $\text{Bern}(\frac{n}{N})$  random variables. We may rewrite the definition of  $\psi_0$  from (3) using these  $Y_i$  random variables as

$$\psi_0 = |\mathcal{H}| \cdot \mathbb{E} \exp \left( \lambda \sum_{i=1}^n Y_i - 16\lambda t \right).$$

To prove the claim, we must show that this is less than 1.

To do so, consider any fixed  $h \in \mathcal{H}$ . We will use the Chernoff bound as stated in Theorem 11, with  $1 + \delta = 16tN/n^2$ . (Note that  $\delta \geq 1$ , as required, since  $N \geq n^3$ .) The value of  $\lambda$  required by the theorem is  $\ln(1 + \delta) = \ln(16tN/n^2)$ , which matches the definition in Algorithm 1. Lastly, note that

$$\mu = \mathbb{E} \sum_{i=1}^n Y_i = n^2/N,$$

since each  $Y_i$  is  $\text{Bern}(n/N)$ . Thus  $\lambda(1 + \delta)\mu = 16\lambda t$ . Applying the theorem, we obtain

$$\mathbb{E} \exp \left( \lambda \sum_{k=1}^n Y_i - 16\lambda t \right) \leq (1 + \delta)^{-(1+\delta)\mu/4} = (16tN/n^2)^{-4t} < n^{-4t} \leq N^{-t},$$

since  $n^3 \leq N \leq 2n^3$  by (1), and also using  $n \geq 2$ . Thus, in conclusion

$$\psi_0 < |\mathcal{H}| \cdot N^{-t} = 1. \quad \triangleleft$$

---

**Algorithm 2** Construction of  $\Pi$  from  $\mathcal{G}$ .
 

---

**Input:**  $t$ -independent family  $\mathcal{G}$  of  $[n] \rightarrow \mathbb{Z}_N$  maps.

**Output:**  $t$ -rankwise independent family of permutations on  $[n]$ .

```

1: Let  $\Pi \leftarrow \emptyset$ 
2: Let  $\tau \leftarrow 32t$ 
3: for  $g \in \mathcal{G}$  do
4:   Let  $\Sigma = \{ (\sigma_1, \dots, \sigma_N) : \sigma_i \text{ is a permutation of } g^{-1}(i) \}$ 
5:   Let  $s \leftarrow \tau! / |\Sigma|$ 
6:   for  $(\sigma_1, \dots, \sigma_N) \in \Sigma$  do
7:     Let  $L \leftarrow []$  be an empty list
8:     for  $i = 1, \dots, N$  do
9:       Append to  $L$  the elements of  $g^{-1}(i)$  in the order given by  $\sigma_i$ 
10:    end for
11:    Add  $s$  copies of the permutation  $\pi : [n] \rightarrow [n]$ , where  $\pi(i) = L[i]$ , to the set  $\Pi$ 
12:  end for
13: end for
14: return  $\Pi$ 
    
```

---

## 2.4 Construction of $\Pi$

The last step is to use the family  $\mathcal{G}$  of maps to build the  $t$ -rankwise independent family  $\Pi$  of permutations on  $[n]$ . Pseudocode for this process is shown in Algorithm 2. Roughly speaking, the algorithm first sorts the elements of  $[n]$  according to the order induced by the functions in  $\mathcal{G}$  and then “breaks ties” using permutations in  $\Sigma$  (see line 4); also note that the number of new permutations will hence depend on  $|\Sigma|$  which is not necessarily fixed for all  $g \in \mathcal{G}$ . The algorithm finally inserts the new permutations in  $\Pi$ . Note that in the algorithm, we view integers  $i \in [N]$  as elements of  $\mathbb{Z}_N$  in the natural manner.

In order for line 11 to make sense, we must establish the following claim.

▷ **Claim 12.** The value  $s = \tau! / |\Sigma|$  is a positive integer.

*Proof.* As above, define

$$\begin{aligned} \mathcal{Y} &= \{ y \in \mathbb{Z}_N : |g^{-1}(y)| \geq 2 \} \\ \mathcal{X} &= \bigcup_{y \in \mathcal{Y}} g^{-1}(y) = g^{-1}(\mathcal{Y}). \end{aligned}$$

Informally,  $\mathcal{Y}$  is the set of bins containing multiple balls, and  $\mathcal{X}$  is the set of balls that are not alone in their bin. By Lemma 7, we know that  $|\mathcal{X}| \leq 32t = \tau$ .

Let  $S_K$  denote the symmetric group on the set  $K$ . Observe that  $\Sigma$  is simply the direct product  $\prod_{y \in \mathbb{Z}_N} S_{g^{-1}(y)}$ , which has an obvious isomorphism to  $\prod_{y \in \mathcal{Y}} S_{g^{-1}(y)}$ , since we can ignore  $y$  with  $|g^{-1}(y)| \in \{0, 1\}$ . In turn, this is isomorphic to a subgroup of  $S_{\mathcal{X}}$ . It follows that  $|\Sigma|$  divides  $|S_{\mathcal{X}}|$ , which divides  $\tau!$  since  $|\mathcal{X}| \leq \tau$ . ◁

▷ **Claim 13.** The family  $\Pi$  is  $t$ -rankwise independent.

*Proof.* We want to show

$$\mathbb{P}(\pi(x_1) < \dots < \pi(x_t)) = \frac{1}{t!} \tag{6}$$

for any  $t$  distinct indices  $x_1, \dots, x_t$ . For notational convenience, let us assume  $x_1 = 1, x_2 = 2, \dots, x_t = t$ . It can be seen that our proof does not use the indices  $x_1, \dots, x_t$ .



To generate  $\pi$ , we will first pick  $g \in \mathcal{G}$  uniformly at random, then pick  $(\sigma_1, \dots, \sigma_N) \in \Sigma$  uniformly at random. Since each  $g \in \mathcal{G}$  produces exactly  $\tau!$  elements in  $\Pi$ , this is equivalent to picking  $\pi$  uniformly. Note that, since  $\Sigma$  is a Cartesian product, the distribution on the  $\sigma_i$  is equivalent to picking  $\sigma_i \in S_{g^{-1}(i)}$  uniformly and independently at random.

For  $i \in [t]$  define

$$R_i = \text{rank of } \pi(i) \text{ among } \pi(1), \dots, \pi(t) = |\{j \in [t]: \pi(j) \leq \pi(i)\}|.$$

Let  $\bar{R} = (R_1, \dots, R_t)$ . Let us view  $\bar{R}$  as an element of the symmetric group  $S_t$  (with  $\bar{R}(i) = R_i$ ). In the remainder of the proof, we will establish that

$$\mathbb{P}(\bar{R} = r) = \mathbb{P}(\bar{R} = r\rho) \quad \forall r, \rho \in S_t. \tag{7}$$

Together with the fact that  $1 = \sum_{\rho \in S_t} \mathbb{P}(\bar{R} = r\rho)$ , we obtain  $\mathbb{P}(\bar{R} = r) = \frac{1}{t!} \forall r \in S_t$ . Thus, when  $r$  is the identity permutation, this establishes (6), for the case  $x_i = i \forall i \in [t]$ .

In order to prove (7), let us introduce some notation for convenience. Throughout the proof, let  $\bar{X}$  denote the random vector  $(X_1, X_2, \dots, X_t)$  where  $X_i = g(i)$ . Let  $\bar{i}$  denote the  $t$ -tuple  $\bar{i} = (i_1, \dots, i_t) \in \mathbb{Z}_N^t$ . Intuitively,  $X$  gives the random locations of the first  $t$  balls, and  $\bar{i}$  gives a specific list of locations that might be the outcome for those balls.

By the law of total probability

$$\mathbb{P}(\bar{R} = r) = \sum_{\bar{i} \in \mathbb{Z}_N^t} \mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}) \cdot \mathbb{P}(\bar{X} = \bar{i}) \tag{8}$$

$$\mathbb{P}(\bar{R} = r\rho) = \sum_{\bar{i} \in \mathbb{Z}_N^t} \mathbb{P}(\bar{R} = r\rho \mid \bar{X} = \bar{i}) \cdot \mathbb{P}(\bar{X} = \bar{i}) \tag{9}$$

Since  $\rho$  is a permutation, one can write the second equation as

$$\mathbb{P}(\bar{R} = r\rho) = \sum_{\bar{i} \in \mathbb{Z}_N^t} \mathbb{P}(\bar{R} = r\rho \mid \bar{X} = \bar{i}\rho) \cdot \mathbb{P}(\bar{X} = \bar{i}\rho), \tag{10}$$

where, for a  $t$ -tuple  $v$  and permutation  $\rho \in S_t$ , the notation  $v\rho$  denotes the  $t$ -tuple whose coordinates are permuted according to  $\rho$ , i.e.,  $(v\rho)_i = v_{\rho(i)}$ .

Observe that by the  $t$ -independence of  $X_1, \dots, X_t$ , we have

$$\mathbb{P}(\bar{X} = \bar{i}) = \mathbb{P}(\bar{X} = \bar{i}\rho) = \frac{1}{N^t}.$$

Thus to show (8) equals (10), it suffices to show that

$$\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}) = \mathbb{P}(\bar{R} = r\rho \mid \bar{X} = \bar{i}\rho).$$

Call the permutation  $r \in S_t$  “feasible” w.r.t. the sequence  $i_1, \dots, i_t$  if for any  $p, q \in [t]$ , if  $i_p < i_q$  then  $r(p) < r(q)$ . In words, this means that the order of  $i_1, \dots, i_t$  is given by the permutation  $r$ . It is possible that several indices in  $[t]$  have the same value in the sequence  $i_1, \dots, i_t$ , in which case  $r$  is allowed to induce any ordering among them.

We observe that  $\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}) = 0 \iff r$  is not feasible w.r.t  $\bar{i}$ . We also note that  $r$  is feasible w.r.t  $\bar{i}$  iff  $r\rho$  is feasible w.r.t  $\bar{i}\rho$ , and hence

$$\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}) = 0 \iff \mathbb{P}(\bar{R} = r\rho \mid \bar{X} = \bar{i}\rho) = 0.$$

So it remains to check the equality of the conditional probabilities for a permutation  $r$  feasible to the  $t$ -tuple  $\bar{i}$ . In fact we can calculate the conditional probability explicitly.

## 67:10 Explicit and Near-Optimal Construction of $t$ -Rankwise Independent Permutations

Let  $S = \{i_1, \dots, i_t\}$  and for  $s \in \mathbb{Z}_N$ , let  $B_s = \{k \in [t] : i_k = s\} \subseteq g^{-1}(s)$  (observe that  $B_s = \emptyset \forall s \notin S$ ). If one views the indices  $[t]$  as balls being thrown into the bins  $\mathbb{Z}_N$ , then  $S$  would be the set of bins occupied by  $[t]$  and  $B_s$  represents balls among  $[t]$  falling into bin  $s$ . For  $s \in \mathbb{Z}_N$  define the event

$$E_s = \{ \forall i, j \in B_s, \sigma_s(i) < \sigma_s(j) \iff r(i) < r(j) \} = \{ \sigma_s \text{ permutes } B_s \text{ according to } r \}.$$

Note that the permutation  $\sigma_s$  is chosen uniformly at random from  $S_{g^{-1}(s)}$ , and hence there is  $\frac{1}{|B_s|!}$  probability that the rank induced over the indices appearing in  $B_s$  is the same rank as the one induced by  $r$ . That is,

$$\mathbb{P}(E_s \mid \bar{X} = \bar{i}) = \frac{1}{|B_s|!}.$$

Note that assuming  $r$  is feasible w.r.t  $\bar{i}$ , we have  $\bar{R} = r$  iff  $\bar{R}$  and  $r$  induce the same order over all the entries of  $B_s$  for all  $s \in S$ . That is,

$$\{\bar{R} = r\} = \bigcap_{s \in S} E_s$$

conditioned on  $\bar{X} = \bar{i}$ .

Note that the permutations  $\{\sigma_s : s \in S\}$  are chosen independently when conditioned on  $\bar{X} = \bar{i}$  so  $\{E_s\}_{s \in S}$  are independent and hence

$$\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}) = \mathbb{P}\left(\bigcap_s E_s \mid \bar{X} = \bar{i}\right) = \prod_{s \in S} \mathbb{P}(E_s \mid \bar{X} = \bar{i}) = \prod_{s \in S} \frac{1}{|B_s|!}.$$

Finally, we verify that the analogous computation for  $\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}\rho)$  yields the same result. Let  $S' = \{(\bar{i}\rho)_k : k \in [t]\}$ ; since  $\rho$  is a permutation, it follows that  $S' = S$ . Similarly letting  $B'_s = \{k \in [t] : (\bar{i}\rho)_k = s\}$ , this time we have

$$\mathbb{P}(\bar{R} = r \mid \bar{X} = \bar{i}\rho) = \prod_{s \in S' = S} \frac{1}{|B'_s|!}.$$

However it is clear that  $|B_s| = |B'_s| \forall s \in \mathbb{Z}_N$ , as  $B'_s = (\rho^{-1})(B_s)$  (since  $\rho^{-1}$  is a bijection between the two sets). Therefore

$$\prod_{s \in S'} \frac{1}{|B'_s|!} = \prod_{s \in S} \frac{1}{|B_s|!}$$

which we argued earlier is sufficient to prove (7). ◁

▷ **Claim 14.** There is a constant  $C > 0$  such that  $|\Pi| \leq (Cn)^{35t}$ .

*Proof.* It is clear that each map  $g \in \mathcal{G}$  contributes exactly  $|\Sigma| \cdot s = \tau!$  permutations to  $\Pi$ . Thus,

$$|\Pi| = \tau! \cdot |\mathcal{G}| \leq (32t)^{32t} \cdot |\mathcal{H}| \leq (32n)^{32t} \cdot N^t \leq (32n)^{32t} \cdot (2n^3)^t,$$

by (1). ◁

### 3 Conclusion and Future Work

Our algorithm for constructing  $\Pi$  runs in time  $n^{O(t)}$ , which is quite efficient size  $|\Pi| = n^{O(t)}$ . However, in applications often one is interested in sampling only a single permutation from  $\Pi$ . In this case, it may be unnecessary to construct the whole family. It is natural to ask if one can give a more explicit construction of  $t$ -rankwise independent families. That is, can a  $t$ -rankwise independent family  $\Pi$  of permutations of  $[n]$  be constructed such that

- $|\Pi| \leq n^{O(t)}$ , and
- sampling a single permutation from  $\Pi$  can be done in time  $O(n)$ ?

We also re-emphasize that the problem of explicitly constructing a  $t$ -independent permutation family  $\Pi$  over  $[n]$  with  $|\Pi| \leq n^{O(t)}$  remains open. Such a construction would strengthen the results of this paper, as it would be a  $t$ -rankwise independent permutation family as well.

---

#### References

- 1 Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000. doi:10.1006/jcss.1999.1690.
- 2 J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979. doi:10.1016/0022-0000(79)90044-8.
- 3 Venkat Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*, 2018. Manuscript.
- 4 Nicholas Harvey and Arvin Sahami. Explicit orthogonal arrays and universal hashing with arbitrary parameters. In *Proceedings of the ACM Symposium on Theory of Computation (STOC)*, 2024.
- 5 Toshiya Itoh, Yoshinori Takei, and Jun Tarui. On permutations with limited independence. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '00*, pages 137–146, USA, 2000. Society for Industrial and Applied Mathematics.
- 6 Enrico Iurlano. Growth of the perfect sequence covering array number. *Des. Codes Cryptography*, 91(4):1487–1494, December 2022. doi:10.1007/s10623-022-01168-3.
- 7 Greg Kuperberg, Shachar Lovett, and Ron Peled. Probabilistic existence of rigid combinatorial structures. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1091–1106, 2012. doi:10.1145/2213977.2214075.
- 8 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- 9 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. doi:10.1561/04000000010.
- 10 Raphael Yuster. Perfect sequence covering arrays. *Des. Codes Cryptography*, 88(3):585–593, March 2020. doi:10.1007/s10623-019-00698-7.

### A Omitted proofs

**Proof of Corollary 8.** For notational convenience, let  $X_i = |g^{-1}(i)|$  for  $i \in [N]$ . Observe that  $n = \sum_{i \in [N]} X_i$  and  $|g([n])| = \sum_{i \in [N]} 1_{\{X_i \geq 1\}}$ . Then we may write

$$\begin{aligned} 2 \cdot (n - |g([n])|) &= 2 \sum_{i \in [N]} \underbrace{(X_i - 1_{\{X_i \geq 1\}})}_{=0 \text{ if } X_i \in \{0, 1\}} = \sum_{i \in [N]} \underbrace{1_{\{X_i \geq 2\}} \cdot 2(X_i - 1)}_{\geq 1_{\{X_i \geq 2\}} \cdot X_i} \\ &\geq \sum_{i \in [N]} 1_{\{X_i \geq 2\}} \cdot X_i = |\mathcal{X}|. \end{aligned}$$

Thus, by Lemma 7,  $|\mathcal{X}| \leq 2 \cdot (n - |g([n])|) \leq 2 \cdot (16t) = 32t$ . ◀

## 67:12 Explicit and Near-Optimal Construction of $t$ -Rankwise Independent Permutations

**Proof of Theorem 11.** Observe that

$$\mathbb{1}_{\{\sum_{i=1}^n Y_i \geq (1+\delta)\mu\}} \leq \exp\left(\lambda \sum_{i=1}^n Y_i - \lambda(1+\delta)\mu\right)$$

and hence taking expectations implies

$$\mathbb{E}\mathbb{1}_{\{\sum_{i=1}^n Y_i \geq (1+\delta)\mu\}} = \mathbb{P}\left(\sum_{i=1}^n Y_i \geq (1+\delta)\mu\right) \leq \mathbb{E}\exp\left(\lambda \sum_{i=1}^n Y_i - \lambda(1+\delta)\mu\right).$$

Next, as shown in [8, Theorem 4.1 and its proof], letting  $\lambda = \ln(1+\delta)$ , we have the inequality

$$\mathbb{E}\exp\left(\lambda \sum_{i=1}^n Y_i - \lambda(1+\delta)\mu\right) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu.$$

It remains to prove that

$$\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \leq (1+\delta)^{-(1+\delta)\mu/4} \quad \forall \delta \geq 1.$$

As  $0 \leq \mu$ , it suffices to show

$$\frac{e^\delta}{(1+\delta)^{1+\delta}} \leq (1+\delta)^{-(1+\delta)/4} \quad \forall \delta \geq 1.$$

After taking logs and performing simple algebraic manipulations, we arrive at another equivalent inequality

$$\frac{4}{3} \leq \left(1 + \frac{1}{\delta}\right) \ln(1+\delta) \quad \forall \delta \geq 1.$$

For  $x \geq 0$ , let  $f(x) = \left(1 + \frac{1}{x}\right) \ln(1+x)$ . We note that

$$f'(x) = \frac{x - \ln(1+x)}{x^2} \geq 0 \quad \forall x > 0$$

since  $\ln(x+1) \leq x \quad \forall x > 0$ . Thus in particular  $f$  is non-decreasing over  $[1, \infty)$  and hence

$$\left(1 + \frac{1}{\delta}\right) \ln(1+\delta) = f(\delta) \geq f(1) = 2 \ln(2) > \frac{4}{3} \quad \forall \delta \geq 1$$

as desired. ◀

# Sparse High Dimensional Expanders via Local Lifts

Inbar Ben Yaacov   

Weizmann Institute of Science, Rehovot, Israel

Yotam Dikstein   

Institute for Advanced Study, Princeton, NJ, USA

Gal Maor  

Tel Aviv University, Tel Aviv, Israel

---

## Abstract

---

High dimensional expanders (HDXs) are a hypergraph generalization of expander graphs. They are extensively studied in the math and TCS communities due to their many applications. Like expander graphs, HDXs are especially interesting for applications when they are bounded degree, namely, if the number of edges adjacent to every vertex is bounded. However, only a handful of constructions are known to have this property, all of which rely on algebraic techniques. In particular, no random or combinatorial construction of bounded degree high dimensional expanders is known. As a result, our understanding of these objects is limited.

The degree of an  $i$ -face in an HDX is the number of  $(i + 1)$ -faces that contain it. In this work we construct complexes whose higher dimensional faces have bounded degree. This is done by giving an elementary and deterministic algorithm that takes as input a regular  $k$ -dimensional HDX  $X$  and outputs another regular  $k$ -dimensional HDX  $\widehat{X}$  with twice as many vertices. While the degree of vertices in  $\widehat{X}$  grows, the degree of the  $(k - 1)$ -faces in  $\widehat{X}$  stays the same. As a result, we obtain a new “algebra-free” construction of HDXs whose  $(k - 1)$ -face degree is bounded.

Our construction algorithm is based on a simple and natural generalization of the expander graph construction by Bilu and Linial [12], which build expander graphs using lifts coming from edge signings. Our construction is based on *local lifts* of high dimensional expanders, where a local lift is a new complex whose top-level links are lifts of the links of the original complex. We demonstrate that a local lift of an HDX is also an HDX in many cases.

In addition, combining local lifts with existing bounded degree constructions creates new families of bounded degree HDXs with significantly different links than before. For every large enough  $D$ , we use this technique to construct families of bounded degree HDXs with links that have diameter  $\geq D$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Expander graphs and randomness extractors

**Keywords and phrases** High Dimensional Expanders, HDX, Spectral Expansion, Lifts, Covers, Explicit Constructions, Randomized Constructions, Deterministic Constructions

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.68

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2405.19191> [11]

**Funding** *Inbar Ben Yaacov:* 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. 819702) and from the Simons Foundation Collaboration on the Theory of Algorithmic Fairness.

*Yotam Dikstein:* This material is based upon work supported by the National Science Foundation under Grant No. DMS-1926686.

*Gal Maor:* Supported by Gil Cohen’s ERC grant 949499.



© Inbar Ben Yaacov, Yotam Dikstein, and Gal Maor;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 68; pp. 68:1–68:24



Leibniz International Proceedings in Informatics

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

## 1 Introduction

Expander graphs are graphs that are well connected. These objects are studied extensively in computer science and mathematics [40], and since their discovery they have found numerous applications in complexity [25, 62], coding theory [64, 66, 26], derandomization [40, 33] and more. Most of these applications rely on families of expander graphs that have a bounded degree. It is well known that random regular graphs are expanders, and many explicit bounded degree constructions are also in hand [55, 51, 63, 12, 54].

Recently, the study of *high dimensional expanders* (HDXs) emerged (see surveys [49, 37]). These are hypergraph analogues of expander graphs. While the full potential of high dimensional expanders is yet to be discovered, they are already important objects of study. High dimensional expanders, and especially bounded degree high dimensional expanders<sup>1</sup> have already yielded exciting applications in various areas such as locally testable codes [26, 61, 28], quantum complexity [7], sampling and Markov chains [27, 43], agreement testing [27, 24, 8], high dimensional geometry and topology [38, 30], pseudorandomness [19] and random (hyper)graph theory [46, 56].

The specific family of high dimensional expanders used in many of the aforementioned applications is tailor-made to satisfy other desired properties, in addition to high dimensional expansion. For example, the local neighborhoods in the high dimensional expanders used in [26, 61, 28] are tailored so that one can define a small locally testable code on them; the high dimensional expanders in [38, 24, 8] also have a vanishing 1-cohomology over certain group coefficients.

However, constructing bounded degree high dimensional expanders (for arbitrarily small spectral expansion of the links) is still a serious challenge. No random model for bounded degree high dimensional expanders is known, and all deterministic constructions known use non-trivial algebraic techniques. The fact that we have only a handful of bounded degree constructions to choose from, makes these objects difficult to understand and to work with. We believe that many further applications await us once we learn how to diversify these constructions, in the same way that many of the above-mentioned applications of expander graphs grew out of more varied expander constructions that were discovered.

Nowadays, all known constructions of HDXs rely on algebraic techniques, including quotients of the Bruhat Tits buildings [10, 16, 45, 52, 22, 24, 8] and coset complexes [42, 31, 59] (see also [39] for a more elementary analysis of some of these HDXs). There have been attempts at constructing bounded degree HDXs with combinatorial tools, but all these constructions fall short either in bounded degree [35, 47] or in their local spectral expansion [20, 21, 17, 48, 34].

In particular, it is an important open question whether an algorithm à la Zig-Zag product [63] exists for bounded degree high dimensional expanders. That is, an algorithm that given a bounded degree high dimensional expander as input, outputs another high dimensional expander with more vertices and the same bound on the degree and spectral expansion.

As an intermediate result, in this work we develop an algorithm that takes a high dimensional expander as input, and outputs another high dimensional expander with more vertices, the same bound on spectral expansion and the same bound on *the degree of high dimensional faces* (but not on the degree vertices). This algorithm is entirely combinatorial, relying only on the theory of graph covers initiated by [5, 12]. While families of complexes

---

<sup>1</sup> A family of HDXs is called *bounded degree* if there is some  $M > 0$  so that all vertices in every HDX in the family have degree at most  $M$ .

constructed via such an algorithm are not sufficient for applications that require the vertex degree to be bounded, we view this as a stepping stone towards an “algebra free” construction of bounded degree HDXs. One exception to this is the recent work by [3], which analyzes a variant of the well known Glauber dynamics (or up-down) random walk on HDXs. The walk that [3] analyzes has bounded degree if and only if the underlying HDX is  $(k - 1)$ -bounded degree.

## 1.1 Preliminaries on High Dimensional Expanders

To better understand our results, let us introduce some standard definitions and notation on simplicial complexes (see Section 2.3 for a more elaborated definitions). A simplicial complex is a hypergraph that is downwards closed to containment. A simplicial complex is  $k$ -dimensional if the largest hyperedge in the complex is of size  $(k + 1)$ . We denote by  $X(\ell)$  the sets (aka faces) of size  $\ell + 1$ . Let  $X$  be a  $k$ -dimensional simplicial complex.

The degree of a face  $\sigma \in X(\ell)$ , is  $d(\sigma) := |\{\tau \in X(\ell + 1) \mid \tau \supseteq \sigma\}|$ . A family of complexes  $\{X_i\}_{i=0}^\infty$  is  $\ell$ -bounded degree if there exists an  $M > 0$  that bounds the degrees of *all*  $\ell$ -faces across *all the complexes simultaneously*. That is, for every  $i$  and any  $\sigma \in X_i(\ell)$ ,  $d(\sigma) \leq M$ . We say that a family of complexes are bounded degree if they are 0-bounded degree. We say that a complex is  $(d_0, d_1, \dots, d_{k-1})$ -regular if for every  $\sigma \in X(\ell)$ ,  $d(\sigma) = d_\ell$ .

In this paper we are mainly interested in the local spectral expansion definition of high dimensional expanders (see [49] for a survey on other definitions). For this we need to define “links”, the generalization of vertex neighborhoods in graphs. For a face  $\sigma \in X$ , the link of  $\sigma$  is the simplicial complex  $X_\sigma = \{\tau \setminus \sigma \mid \sigma \subseteq \tau \in X\}$ . We will be interested in the graph structure underlying the complex and its links. The 1-skeleton of  $X$  is the graph whose vertices are  $X(0)$  and whose edges are  $X(1)$ .

► **Definition 1** (High dimensional expander). *For  $\lambda > 0$  we say that  $X$  is a  $\lambda$ -two sided (one sided) high dimensional expander if for every  $\ell \leq k - 2$  and  $\sigma \in X(\ell)$ , the 1-skeleton of  $X_\sigma$  is a  $\lambda$ -two sided (one sided) spectral expander.*

## 1.2 Our Results

Our results are based on the notion of a graph lifts. We say a graph  $\widehat{G} = (\widehat{V}, \widehat{E})$  is a lift of a graph  $G = (V, E)$  if there exists a graph homomorphism  $\phi : \widehat{V} \rightarrow V$  such that for every  $\hat{v} \in \widehat{V}$ , the mapping  $\phi$  is a bijection on the neighborhood of  $\hat{v}$ . Intuitively, a lift of a graph  $G$  is a large graph  $\widehat{G}$  that locally looks the same as  $G$ . Graph lifts are essential in many constructions of expander graphs [5, 12, 54], and our construction builds on the beautiful work of [12]. We elaborate more on this below.

Our main result is a construction algorithm that maintains both expansion and the degree of the  $(k - 1)$ -faces of a regular complex. This algorithm takes as input a  $(d_0, d_1, \dots, d_{k-1})$ -regular  $\lambda$ -high dimensional expander  $X$ , and outputs a  $(2d_0, 2d_1, \dots, 2d_{k-2}, d_{k-1})$ -regular  $\widehat{X}$  with twice as many vertices that is also a  $\lambda$ -HDX (even though the number of intermediate faces grows like  $|\widehat{X}(i)| = 2^i |X(i)|$  for  $i \leq k - 1$ ). This algorithm uses the notion of random lifts [12], and in particular, it requires no algebraic machinery for the construction nor the analysis. More formally, this is the theorem we prove.

► **Theorem 2** (See Theorem 28 for a more precise statement). *There exists a randomized algorithm  $\mathcal{A}$  that takes as input a  $k$ -dimensional complex  $X_0$  and an integer  $i \geq 1$ , runs in expected time  $\text{poly}((2^i |X_0(0)|)^k)$  at most, and outputs a  $k$ -dimensional complex  $X_i$  with  $2^i |X_0(0)|$  vertices. The algorithm has the following guarantees.*

1. If  $X_0$  is a  $(d_0, d_1, \dots, d_{k-1})$ -regular  $\lambda$ -two sided high dimensional expander, then  $X_i$  is a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided high dimensional expander where  $\lambda' = O\left(\max\left\{\lambda\left(1 + \log \frac{1}{\lambda}\right), \sqrt{\frac{\log^3 d_{k-1}}{d_{k-1}}}\right\}\right)$ .
2. For every  $\hat{\sigma} \in X_i(k-2)$ , the link  $(X_i)_{\hat{\sigma}}$  is a lift of  $(X_{i-1})_{\sigma}$  for some  $\sigma \in X_{i-1}(k-2)$ . For every  $j \leq k-2$ ,  $|X_i(j)| = 2^{(j+1)i}|X_0(j)|$ .

There are various complexes in hand that one can use as the input to this algorithm. These include the complete complex, the complexes from [50], and even complexes from bounded degree families that are regular, such as those constructed by [31].

We give two proofs to Theorem 2, building on the techniques developed by [12] to analyze lifts in expander graphs, and extend them to high dimensional expanders.

While most of the work in [12] regards random lifts of graphs, they also show how to deterministically find expander graphs using lifts. Building on their method, we also give a deterministic algorithm for finding the complexes in Theorem 2, albeit under some more assumptions on the input  $X_0$ . This provides a *deterministic, polynomial time and elementary* construction of a family of  $(k-1)$ -bounded  $k$ -dimensional high dimensional expanders.

► **Theorem 3** (See Theorem 35). *There exists a deterministic algorithm  $\mathcal{B}$  that takes as input a  $k$ -dimensional complex  $X_0$  and an integer  $i \geq 1$ , runs in time  $\text{poly}((2^i |X_0(0)|)^k)$  at most, and outputs a  $k$ -dimensional complex  $X_i$  with  $2^i |X_0(0)|$  vertices. The algorithm has the following guarantees.*

1. If  $X_0$  is a  $(d_0, d_1, \dots, d_{k-1})$ -regular  $\lambda$ -two sided high dimensional expander, such that  $d_{k-1} > 2^{10k}$  and  $|X_0(k-2)| \leq (d_{k-2})^{10k}$ , then  $X_i$  is a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided high dimensional expander where  $\lambda' = O\left(2^{5k} \max\left\{\lambda\left(1 + \log \frac{1}{\lambda}\right), \sqrt{\frac{\log^3 d_{k-1}}{d_{k-1}}}\right\}\right)$ .
2. For every  $\hat{\sigma} \in X_i(k-2)$ , the link  $(X_i)_{\hat{\sigma}}$  is a lift of  $(X_{i-1})_{\sigma}$  for some  $\sigma \in X_{i-1}(k-2)$ . For every  $j \leq k-2$ ,  $|X_i(j)| = 2^{(j+1)i}|X_0(j)|$ .

Not only is our construction deterministic, but it is also simple and versatile; one can apply it to various kinds of high dimensional expanders, and the family of HDXs obtained by doing so is changes according to the initiating HDX given at the beginning of the process.

### 1.3 Comparing to Random Constructions of HDXs

In the graph case the configuration model yields regular and bounded degree expanders. In contrast, there is no immediate generalization of this model to higher dimensions, that leads to bounded degree HDXs, even if one only wishes to bound the degrees of higher dimensional faces. If one settles for logarithmic degree, then one could use the [46] random model to construct random HDXs. The degree of the top-level faces of these complexes is  $O(\log n)$ , where  $n$  is the number of vertices, and the degree of the lower dimensional faces is polynomial in  $n$ . For 2-dimensional complexes, the random geometric model in [47] offers an improvement to the vertex degree that one gets from [46], but it is still polynomial.

It is tempting to try and adapt the [46] model for constructing  $(k-1)$ -bounded degree HDXs, but doing so in a straightforward manner falls short of achieving that. The work by [50] found an appropriate generalization of the random model that gives  $(k-1)$ -bounded degree HDXs, utilizing the breakthrough work of [44] on Steiner systems. In their model, one takes a complete  $(k-1)$ -skeleton and samples  $k$ -faces by sampling random Steiner systems on this complex.

Our construction sidesteps this difficulty by taking a different approach; it uses random *local lifts* of HDXs (presented in the following subsection) instead of trying to construct random ones from scratch. In this setting, the high dimensional case behaves more similar



to the 1-dimensional one - our work shows that random local lifts of HDXs are HDXs with high probability. Of course, this requires an appropriate modification of the lift notion to local lifts.

### 1.4 The Construction

We now dive into the heart of Theorem 2. Our construction builds upon the work of random lifts of graphs studied in [12]. Random lifts of expanders have been subject to extensive research (e.g., [6, 12, 2, 54, 14]). However, in this work, we do not try to lift the complex itself. Instead, we construct another complex where the  $(k - 2)$ -links are lifts of links in the original complex. We call such a complex a *local lift*.

Let us first explain the idea behind the work of [12]. Their work suggests a construction of bounded degree family of expander graphs  $\{G_i\}_{i=0}^\infty$ , where for every  $i$ ,  $G_{i+1}$  is a lift of  $G_i$ . The fact that the maximal degree of  $G_{i+1}$  is equal to the maximal degree of  $G_i$  promises that the sequence is bounded degree. Therefore, one only needs to worry about expansion.

The work [12] studies random lifts sampled using signings on the edges of a graph  $G = (V, E)$ , that is, functions  $f : E \rightarrow \{\pm 1\}$ . Given such a signing  $f$ , one can construct the following lift  $\widehat{G} = (\widehat{V}, \widehat{E})$  by setting  $\widehat{V} = V \times \{\pm 1\}$  and  $\{(v, i), (u, j)\} \in \widehat{E}$  if  $\{v, u\} \in E$  and  $i \cdot j = f(\{u, v\})$ .

The work of [12] analyzes when a lift  $\widehat{G}$  obtained by random signing  $f$  is an expander graph. They give a proof (based on the Lovász Local Lemma) that every expander  $G$  has such a “good” signing. They also provide a deterministic algorithm to construct such a lift using the conditional probabilities method [4]. Our construction generalizes this idea, only instead of lifts coming from edge signings, we define *local lifts* coming from *face*-signings.

Let  $X$  be a  $k$ -dimensional simplicial complex and let  $f : X(k) \rightarrow \{\pm 1\}$ . Define the  $k$ -dimensional complex  $\widehat{X}$  (where  $f$  is implicit in the notation) as a complex whose vertices are  $\widehat{X}(0) = X(0) \times \{\pm 1\}$ , and whose  $k$ -faces are

$$\widehat{X}(k) = \left\{ \left\{ v_0^{j_0}, v_1^{j_1}, \dots, v_k^{j_k} \right\} \mid \{v_0, v_1, \dots, v_k\} \in X(k) \text{ and } \prod_{i=0}^k j_i = f(\{v_0, v_1, \dots, v_k\}) \right\}.$$

For  $1 \leq \ell \leq k - 1$  the  $\ell$ -faces are independent of the second coordinate, that is,

$$\widehat{X}(\ell) = \left\{ \left\{ v_0^{j_0}, v_1^{j_1}, \dots, v_\ell^{j_\ell} \right\} \mid \{v_0, v_1, \dots, v_\ell\} \in X(\ell) \right\}.$$

Obviously, the underlying graph of  $\widehat{X}$  is *not* a lift of the underlying graph of  $X$ . Indeed, the degree of each vertex is doubled. However, for every  $\hat{\sigma} \in \widehat{X}(k - 2)$ , we show that  $\widehat{X}_{\hat{\sigma}}$  is isomorphic to a lift of  $X_\sigma$  (where  $\sigma = \{v \mid v^j \in \hat{\sigma}\}$ ).

Indeed, let us assume for simplicity that  $k = 2$ . Consider the link of a vertex  $v^j \in \widehat{X}(0)$  and define the function  $g : X_v(1) \rightarrow \{\pm 1\}$  by  $g(uw) = j \cdot f(uw)$ . We claim that  $\widehat{X}_{v^j}$  is the cover  $g$  induces on  $X_v$ . It is easy to see that its vertices are  $\widehat{X}_{v^j}(0) = X_v(0) \times \{\pm 1\}$ , since the vertices in  $\widehat{X}_{v^j}$  correspond to edges in  $\widehat{X}(1)$ . These are precisely all  $u^1, u^{-1}$  where  $u \in X_v(0)$ .

The edges are more delicate. Edges  $\{u^{j'}, w^{j''}\} \in \widehat{X}_{v^j}(1)$  correspond to triangles  $\{v^j, u^{j'}, w^{j''}\} \in \widehat{X}$ . Indeed, such a triangle is in  $\widehat{X}$  if and only if  $\{u, w\} \in X_v(1)$  and  $j \cdot j' \cdot j'' = f(\{v, u, w\})$ . The second condition occurs if and only if  $j \cdot j' \cdot j'' = g(uw)$ . Hence,  $\widehat{X}_{v^j}$  is the cover  $g$  induces on  $X_v$ .

As mentioned above, we give two proofs that signings  $f$  so that  $\widehat{X}$  is a high dimensional expander exist. The first proof is based on the Lovász Local Lemma and follows the argument in [12, Lemma 3.3], and generalizes it so that multiple links may be taken into account

simultaneously. The second proof is based on a different way to use the Lovász Local Lemma (together with other results from [12]) which we find simpler, to deduce high dimensional expansion. This proof, while more restrictive on the link sizes, can be combined with the algorithmic version of the Lovász Local Lemma [57], to prove Theorem 2. Afterwards, we show that if the links of the complex  $X$  are already dense, then the derandomization technique in [12] works for high dimensional expanders, and we can obtain a deterministic construction for  $(k - 1)$ -bounded  $k$ -dimensional high dimensional expanders, proving Theorem 3.

## 1.5 Understanding Vertex vs. Edge Degree in Bounded-Degree Constructions

We can use Theorem 2 to diversifying links in other existing bounded degree constructions, and thus gain more understanding on how possible high dimensional expanders may look like. For simplicity, let us stick to the 2-dimensional case, and consider the question *how small could  $d_1$  be given  $d_0$  in a  $(d_0, d_1)$ -regular high dimensional expander?*

Let us consider the behavior of  $d_0$  and  $d_1$  in the known bounded degree constructions [10, 16, 45, 52, 42, 31, 22, 24, 8]. In all the above,  $d_0$  grows to infinity as  $\lambda$  goes to 0, and  $d_1 = \text{poly}(d_0)^2$ . In other words, the links themselves are “locally” dense. A natural question to ask is whether the lower bound of  $d_1 \geq d_0^{\Omega(1)}$  is necessary for *bounded degree* constructions. In expander *graphs* it is well known that one can increase the size of the graph without increasing the bound on the degree, but this is not the behavior in the known bounded-degree HDX constructions.

We note that if one allows  $d_0$  to tend to infinity with  $n$ , rather than staying constant, then works such as [50] (and also infinite families of complexes constructed by iteratively applying Theorem 3) show that this is false. But this question is more interesting when its bounded degree.

Theorem 3 gives a negative answer to this question in the 2-dimensional case, by proving the following.

► **Theorem 4.** *For every  $\lambda > 0$  and any sufficiently large  $M > 0$ , there exists an infinite family of 2-dimensional  $\lambda$ -two sided high dimensional expanders that are  $(d_0, \exp(\text{poly}(\frac{1}{\lambda})))$ -regular, for  $M \leq d_0 \leq 2M$ .*

*In particular, for every large enough  $D > 0$ , there exists an infinite family of 2-dimensional  $\lambda$ -two sided HDXs such that the diameter in every link  $X_v$ , is at least  $D$ .*

We stress that  $d_1 = \exp(\text{poly}(\frac{1}{\lambda}))$  depends only on the spectral expansion and not on the number of vertices or  $d_0$ .

The proof of Theorem 4 appears in the full version of this paper [11].

## 1.6 Related Work

### Bounded degree HDX

As discussed above, all known constructions of bounded degree high dimensional expanders use algebraic techniques. The first bounded degree high dimensional expanders for arbitrarily small  $\lambda > 0$  was by [10]. This was followed by many other works that aimed to construct the high dimensional equivalent to Ramanujan graphs [16, 45, 53, 52]. All these constructions

---

<sup>2</sup> Technically most of the constructions above are not *regular*, only bounded degree, so  $d_0$  and  $d_1$  should be average values, but we ignore this point for the sake of presentation.

are quotients of  $\tilde{A}$ -type Bruhat Tits buildings. The work by [22] used this building together with complex lifts to construct other high dimensional expanders. Recently, high dimensional expanders that come from  $\tilde{C}$  Bruhat Tits buildings were also constructed and studied [18, 24, 8]. A second type of constructions come from coset complexes, first studied by [42]. More complexes of this type were constructed by [31, 59]. We mention that the work by [39] simplified the analysis of these coset complexes, and gave a description of the complexes in [42] in relatively elementary means (albeit still relying on algebraic methods).

Interestingly, [50] give a randomized construction of a  $(k - 1)$ -bounded degree family of  $\lambda$ -HDXs for arbitrarily small  $\lambda > 0$ . This construction is based on random Steiner systems and given in the breakthrough result of [44]. The underlying  $(k - 1)$ -skeletons of the complexes in that family are complete.

There are other bounded degree constructions [20, 21, 17, 48, 34]. These constructions have various mixing properties, but none of them are  $\lambda$ -HDXs for  $\lambda < \frac{1}{2}$  (where  $\lambda$  is normalized between 0 and 1). There are other constructions of  $\lambda$ -HDXs for  $\lambda < \frac{1}{2}$ , which are not bounded degree, but are still non-trivially sparse. These include [35] - based on Grassmann posets, and [47] - based on random geometric graphs of the sphere.

Finally, we comment that previous works also considered the possible degrees  $(d_0, d_1)$  possible in a high dimensional expanders. The work by [31] used irregular algebraic constructions of bounded degree  $\lambda$ -HDXs and “regularized” them, thus showing that there exists bounded degree HDXs that are regular for arbitrarily small  $\lambda$ . The work by [17] gives a lower bound on the expansion of the underlying graph of the complex in terms of  $(d_0, d_1)$ , but this lower bound does not rule out (or construct) such HDXs with  $d_1 \ll d_0$ .

## Graph and HDX lifts

The study of random graph lifting was initiated in [5]. Random lifts from signings of expanders were studied in [12] where it was proven that with high probability they are also expanders. This was extended to larger lifts as well [60, 1]. Friedman showed that random lifts of Ramanujan graphs are nearly Ramanujan [32] (see also [14]). In the seminal paper by [54], Ramanujan bipartite graphs were constructed by using graph lifts. One can also define lifts of simplicial complexes. Most known bounded degree high dimensional expanders are constructed using a dual notion of lifts - that is, taking quotients of an infinite object [10, 16, 45, 53, 52, 42], in a way such that the infinite object is a lift of the complex that is constructed. [22] studied taking random lifts of simplicial complexes as in [12], but the construction there needed the use of algebraic techniques as well.

## 1.7 Open Questions

### Combinatorial constructions

As we mention earlier, there is no construction of bounded degree high dimensional expanders that does not rely on non-trivial algebraic techniques. As an intermediate step towards such a construction, can one give a construction of  $k$ -dimensional simplicial complexes that are  $(k - 2)$ -bounded degree (or  $i$ -bounded degree for any  $i < k - 1$ )?

### Links with other properties

Fix a vertex set  $[n]$  and graphs  $\{G_i\}_{i=1}^n$  (one graph for every vertex  $i \in [n]$ ). It is interesting to understand whether there exists a graph whose vertex set is  $V = [n]$ , and such that the neighborhood of every vertex  $i \in V$  is (isomorphic to)  $G_i$ . The structure of such graphs is an

extensive topic of study, especially in the case where all  $G_i$ 's are equal (see, e.g., [15, 13, 58]). One of the major components in the works [26, 61] that constructed asymptotically good locally testable codes and quantum codes, is a construction of graphs that locally look like a neighborhood of a graph product, but globally have improved expansion properties.

In this work, we propose a technique that addresses a related problem. Given a graph  $G$  (which is the one skeleton of a regular 2-dimensional complex  $X$ ), we find a graph  $\widehat{G}$  where every neighborhood in  $\widehat{G}$  is a random (or deterministic) 2-lift of a corresponding neighborhood in  $G$ . Is there a technique that allows us to do so for *any* set of 2-lifts of the respective vertex neighborhoods?

### Other notions of expansion

In this paper we mainly deal with local spectral expansion, but other definitions of high dimensional expansion also exist. Most notable is the notion of coboundary expansion defined independently in [46] and [38]. This notion is important for many applications of high dimensional expanders such as code construction [26], topological expansion [38] and property testing [41, 36, 23, 9]. Does a local lift maintain coboundary expansion? If not, is it maintained in interesting special cases?

### Better local spectral expansion

Works following [12] such as [54, 14] improved the bounds on the spectrum of lifts of regular graphs. Can one construct local lifts of regular high dimensional expanders that are also *Ramanujan*?

## 1.7.1 Organization of This Paper

The necessary preliminaries are given in Section 2. We describe local lifts in Section 3 and describe some of their basic properties. In Section 4 we show existence of good local lifts by modifying a Lovász Local Lemma argument by [12]. In Section 5 we prove Theorem 2 using the algorithmic Lovász Local Lemma [57] and derive Theorem 4. In Section 6 we show that the method of derandomization in [12] could be generalized to our case as well and prove Theorem 3.

## 2 Preliminaries

Unless explicitly stated, all logarithms are with base 2. The  $\ln$  function is a logarithm with base  $e$ . We write  $A \sqcup B$  to denote a disjoint union of sets  $A, B$ . For  $n \geq 0$  we write  $[n] = \{0, 1, \dots, n\}$ . For a square matrix (or equivalently, a linear operator on a finite vector space), we write  $\|A\|$  to denote the operator norm.

### 2.1 Graphs

Let  $G = (V, E)$  be a graph. For  $u, v \in V$  we write  $\Gamma(v)$  for the set of  $v$ 's neighbors in  $G$  and  $u \sim v$  if  $u$  and  $v$  are neighbors. The *indicator vector* of a set  $S \subseteq V$ , denoted by  $\mathbf{1}_S$ , is  $\mathbf{1}_S : V \rightarrow \{0, 1\}$  with  $\mathbf{1}_S(v) = 1 \iff v \in S$ . For two sets  $S, T \subseteq V$  we write  $E_G(S, T)$  for the set of edges in  $G$  between  $S$  and  $T$ . The *graph induced by  $S$  and  $T$*  is  $G' = (S \cup T, E_G(S, T))$ . For a  $d$ -regular graph we denote  $\ell_2(V) = \{f : V \rightarrow \mathbb{R}\}$  endowed with the inner product  $\langle f, g \rangle = \sum_{v \in V} f(v)g(v)$ .

### 2.1.1 Expander Graphs

Expander graphs are graphs with good connectivity properties. There are many equivalent ways to define expanders [40]. In this manuscript we focus on *spectral expansion*.

Let  $G = (V, E)$  be a  $d$ -regular graph. The *random walk matrix* of  $G$  is a matrix  $A \in \mathbb{R}^{V \times V}$  defined by  $A(u, v) = \frac{1}{d}$  if  $u \in \Gamma(v)$  and otherwise 0. Equivalently, it corresponds to the random walk operator  $A : \ell_2(V) \rightarrow \ell_2(V)$  with  $Af(v) = \frac{1}{d} \sum_{u \in \Gamma(v)} f(u)$ . We abuse notation and use  $A$  for both the matrix and the random walk operator it represents. We sometimes denote this operator by  $A_G$  when  $G$  is unclear from the context.

The operator  $A$  is self adjoint with respect to the inner product. Therefore, it has an orthonormal basis of real-valued eigenvectors, where the eigenvalues are denoted by  $1 = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ . We elaborate and write  $\lambda_i(G)$  when the graph in question is unclear from the context. The *spectrum of  $G$*  is the spectrum of its random walk matrix and is denoted by  $\text{Spec}(G)$ .

► **Definition 5** (spectral expander). *For  $\lambda \in [0, 1]$  we say that  $G$  is  $\lambda$ -two sided (resp. one sided) spectral expander (or expander for short) if  $\lambda \geq \max\{\lambda_2, |\lambda_n|\}$  (resp.  $\lambda \geq \lambda_2$ ).*

### 2.1.2 Tensor Product

Let  $G, H$  be any graphs. The *tensor product* of  $G$  and  $H$ , denoted by  $G \otimes H$ , is the graph with vertices  $V(G) \times V(H)$ , and edges  $E(G \otimes H) = \{(a, b)(a', b') \mid \{a, a'\} \in E(G) \text{ and } \{b, b'\} \in E(H)\}$ . The following fact is well known.

► **Fact 6.** *Let  $G, H$  be graphs. If  $H, G$  are  $\lambda, \lambda'$ -two sided spectral expanders respectively, then  $G \otimes H$  is a  $\max\{\lambda, \lambda'\}$ -two sided spectral expander. Moreover, if  $H$  is a  $\lambda$ -two sided spectral expander and  $G$  is a  $\lambda'$ -one sided spectral expander, then  $G \otimes H$  is a  $\max\{\lambda, \lambda'\}$ -one sided spectral expander.*

## 2.2 Graph Lifts

Graph lifts are an important notion, studied first by [5, 6] (although the notion of lifts themselves is a classical notion in algebraic topology known for about a century).

► **Definition 7** (lift). *For finite, connected and simple graphs  $G$  and  $\widehat{G}$ , a lift (also known as a covering map)  $\phi : \widehat{G} \rightarrow G$  is a graph homomorphism with the property that for all  $\hat{v} \in V(\widehat{G})$ ,  $\phi$  maps the neighborhood of  $\hat{v}$  in  $\widehat{G}$  onto the neighborhood of  $\phi(\hat{v})$  in  $G$ . Finally, we say that  $\widehat{G}$  is an  $\ell$ -lift of  $G$  if there exists an  $\ell$ -to-1 covering map  $\phi : \widehat{G} \rightarrow G$ .*

One way to construct a 2-lift is to use a signing function on the edges as follows.

► **Definition 8** (Function induced lift). *Let  $G = (V, E)$  be a graph and let  $f : E \rightarrow \{\pm 1\}$  be a signing. The  $f$ -induced lift  $\widehat{G} = \widehat{G}^f$  is the graph whose vertices are  $\widehat{V} = V \times \{\pm 1\} = \{v^j \mid v \in V, j \in \{\pm 1\}\}$  and whose edges are  $\widehat{E} = \{\{v^j, u^i\} \mid \{v, u\} \in E, ij = f(\{u, v\})\}$ . The lift map is  $\phi(v^j) = v$ .*

It is elementary to prove this construction is indeed a lift, so we omit this proof. It is also easy to show that any 2-lift is an induced lift for some signing  $f : V(G) \rightarrow \{\pm 1\}$ . See [65] for a more general statement and proof.

In Section 3 we generalize the notion of graph lifts to *local lifts* of simplicial complexes.

### 2.2.1 Signing Functions and Lift Expansion

Fix a graph  $G$  and a signing  $f : E(G) \rightarrow \{\pm 1\}$ . In this subsection, we characterize the eigenvalues of an  $f$ -induced lift. For this, we need to define the  $f$ -signing of an adjacency operator. For a  $d$ -regular graph the  $f$ -signing of the adjacency operator is the matrix  $A^f(u, v) = f(u, v) \cdot A(u, v)$  for  $\{u, v\} \in E$  and  $A^f(u, v) = 0$  if  $\{u, v\} \notin E$ .

This signing matrix is closely related to the random walk operator of the lift. In particular, the following is by now classical.

► **Lemma 9.** *Let  $G$  be a  $d$ -regular graph and let  $\widehat{G}$  be an  $f$ -induced 2-lift. Then the eigenvalues of  $A_{\widehat{G}}$  are the union (with multiplicities) of the eigenvalues of  $A$  and those of  $A^f$ . In particular, if  $\|A^f\| \leq \lambda$  and  $G$  is a  $\lambda'$ -two sided (resp. one sided) spectral expander, then  $\widehat{G}$  is a  $\max\{\lambda, \lambda'\}$ -two sided (resp. one sided) spectral expander.*

Using this lemma, [12] gave a criterion for the expansion of the lift graph.

► **Lemma 10** ([12, Lemma 3.3]). *Let  $G, f$  and  $A^f$  be as above and assume that  $G$  is a  $\lambda$ -two sided (resp. one sided) spectral expander with no self loops. Assume that for any pair of disjoint  $S, T \subseteq V(G)$  it holds that  $|\langle \mathbf{1}_S, A^f \mathbf{1}_T \rangle| \leq \alpha \sqrt{|S||T|}$ , then  $\widehat{G}^f$  is a  $\lambda'$ -two sided (resp. one-sided) spectral expander where  $\lambda' = \max\{\lambda, O(\alpha(1 + \log \frac{1}{\alpha}))\}$ .*

We note that there is a nice formula for this inner product, which is  $\langle \mathbf{1}_S, A^f \mathbf{1}_T \rangle = \frac{1}{d} \sum_{(v,u):\{v,u\} \in E(G)} f(u, v) \mathbf{1}_S(v) \mathbf{1}_T(u)$ .

## 2.3 High Dimensional Expanders

► **Definition 11** (simplicial complex). *A  $k$ -dimensional simplicial complex is a finite hypergraph  $X$  that is downwards closed to containment. That is, if  $\tau \in X$  and  $\sigma \subseteq \tau$  then  $\sigma \in X$ .*

We write  $X = X(-1) \sqcup X(0) \sqcup X(1) \sqcup \dots \sqcup X(k)$ , where  $X(\ell) = \{\sigma \in X \mid |\sigma| = \ell + 1\}$  (here  $X(-1) = \{\emptyset\}$  is mainly a formality) and the maximal size of a set  $\sigma \in X$  is  $k + 1$ . We call elements  $\sigma \in X(\ell)$   $\ell$ -faces, and in this case we say that  $X$  is  $k$ -dimensional. In this paper we will always assume the simplicial complex in question is *pure*, that is, that every  $\sigma \in X(\ell)$  contained in some  $\tau \in X(k)$ . In addition, we assume it has no self-loops or multifaces. Namely, every vertex appears at most once in each face and any face appears at most once in  $X$ .

The degree of a face  $\sigma \in X(i)$  is  $d(\sigma) = |\{\tau \in X(i+1) \mid \tau \supseteq \sigma\}|$ . We say that a family of simplicial complexes  $\{X_i\}_{i=0}^\infty$  is  $j$ -bounded degree if there is an integer  $M > 0$  so that for all  $X_i$  and all  $\sigma \in X_i(j)$ ,  $d(\sigma) \leq M$ . If the family is 0-bounded degree, we sometimes just say bounded degree (without the zero).

► **Definition 12** (hyper-regularity). *Let  $d_0 \geq d_1 \geq \dots \geq d_{k-1}$  be positive integers. A  $k$ -dimensional simplicial complex  $X$  is  $(d_0, d_1, \dots, d_{k-1})$ -regular if for any  $i \in \{0, \dots, k-1\}$  and any  $i$ -face  $\sigma$ ,  $d(\sigma) = d_i$ .*

We say that  $X$  is regular if there exists such a tuple so that  $X$  is  $(d_0, d_1, \dots, d_{k-1})$ -regular. In this case we denote by  $d_i(X) = d_i$ .

The  $j$ -skeleton of a simplicial complex  $X$  is the simplicial complex obtained by taking all the  $i$ -faces of  $X$ , for all  $i \leq j$ . The 1-skeleton of a complex is also called an *underlying graph*.

A link is a generalization of a graph neighborhood.

► **Definition 13** (link). *For a  $k$ -dimensional simplicial complex  $X$  and a face  $\sigma \in X$ , the link of  $\sigma$  is the  $(k-1-|\sigma|)$ -dimensional simplicial complex  $X_\sigma = \{\tau \setminus \sigma \mid \tau \in X, \tau \supset \sigma\}$ .*

For  $\ell \leq k-2$  and  $\sigma \in X(\ell)$  we denote by  $A_\sigma$  the random walk operator of the 1-skeleton of  $X_\sigma$ . We often abuse of notation and for a face  $\sigma = \{v_0, \dots, v_\ell\} \in X(\ell)$  write  $\sigma = v_0 \dots v_\ell$ .

Natural analogues of expander graphs to higher dimensions are simplicial complexes where the neighborhoods of the faces are themselves expander graphs. See Section 1 for more context on this important definition.

► **Definition 14** ( $\lambda$ -high dimensional expander). *Let  $\lambda \in [0, 1]$ . A  $k$ -dimensional simplicial complex  $X$  is a  $\lambda$ -two sided (resp. one sided) high dimensional expander if for all  $i \leq k - 2$  and all  $\sigma \in X(i)$ , the 1-skeleton of  $X_\sigma$  is a  $\lambda$ -two sided (resp. one sided) spectral expander.*

## 2.4 The Lovász Local Lemma

The Lovász Local Lemma is a classical result in the probabilistic method.

► **Lemma 15** (Lovász Local Lemma [29]). *Let  $\mathfrak{B} = \{B_1, \dots, B_n\}$  be a finite set of events in some arbitrary probability space. The dependency graph of  $\mathfrak{B}$  is a digraph  $G_{\mathfrak{B}} = (\mathfrak{B}, E)$  so that any event  $B_i \in \mathfrak{B}$  is mutually independent of all the events  $\mathfrak{B} \setminus \Gamma(B_i)$ , where  $\Gamma(B_i)$  is the neighborhood of  $B_i$  in  $G_{\mathfrak{B}}$ .*

*If there exists a real function  $\rho : \mathfrak{B} \rightarrow [0, 1)$  so that*

$$\mathbb{P}[B_i] \leq \rho(B_i) \prod_{B_j \sim B_i} (1 - \rho(B_j)) \tag{2.1}$$

*for any  $B_i \in \mathfrak{B}$ , then with strictly positive probability, none of the events  $B_i$  occur.*

This lemma also has an algorithmic version, first given in the seminal work of [57]. We give below a slightly less general version than the one in [57].

► **Lemma 16** ([57]). *Let  $\Omega$  be a finite set and let  $\mathfrak{P} = (P_1, P_2, \dots, P_m)$  be a tuple of independent random variables supported on  $\Omega^m$ . Let  $\mathfrak{B} = \{B_1, B_2, \dots, B_n\}$  be a finite set of events in the sigma algebra of  $\mathfrak{P}$ . Let the dependency graph and the assignment  $\rho : \mathfrak{B} \rightarrow [0, 1)$  be as in Lemma 15. Then there exists a randomized algorithm that finds an assignment  $p \in \Omega^m$  such that  $p \notin \bigcup_{i=1}^n B_i$ . If one can verify whether  $B_i$  holds in time  $t$ , then the randomized algorithm runs in  $tn \sum_{i=1}^n \frac{\rho(B_i)}{1 - \rho(B_i)}$  expected time.*

The algorithm described in this lemma is simple. The algorithm starts with randomly sampling some  $p \in \Omega^m$ . While there exists some  $B_i$  such that  $p \in B_i$ , the algorithm takes an arbitrary such  $B_i$ , and resamples all the coordinates  $P_j$  that  $B_i$  depends upon. Of course, if the algorithm halts, then  $p \notin \bigcup_{i=1}^n B_i$ . The paper [57] shows that the expected number of times an event  $B_i$  is resampled is at most  $\frac{\rho(B_i)}{1 - \rho(B_i)}$  which explains the runtime of this algorithm.

## 3 Local lifts

This section presents our basic construction, the *local lift* of a complex. We will define this construction formally and describe some of its properties.

► **Construction 17** (Local Lift). *Let  $X$  be a  $k$ -dimensional simplicial complex and let  $f : X(k) \rightarrow \{\pm 1\}$ . The  $f$ -local lift of  $X$  denoted by  $\widehat{X} = \widehat{X}^f$ , is the following  $k$ -dimensional simplicial complex:*

- $\widehat{X}(0) = X(0) \times \{\pm 1\}$  and we denote the vertices by  $\widehat{X}(0) = \{v^j \mid v \in X(0), j \in \{\pm 1\}\}$ .
- For any  $1 \leq \ell \leq k - 1$ ,

$$\widehat{X}(\ell) = \left\{ \{v_0^{j_0}, v_1^{j_1}, \dots, v_\ell^{j_\ell}\} \mid \{v_0, v_1, \dots, v_\ell\} \in X(\ell), j_0, \dots, j_\ell \in \{\pm 1\} \right\}.$$

## 68:12 Sparse High Dimensional Expanders via Local Lifts

- Finally,  $\widehat{X}(k)$  is the set of all faces  $\sigma = \{v_0^{j_0}, v_1^{j_1}, \dots, v_k^{j_k}\}$  so that the face without the signs is  $\{v_0, v_1, \dots, v_k\} \in X(k)$ , and the product of the  $j_i$ 's are equal to  $f(\sigma)$ . Namely,

$$\widehat{X}(k) = \left\{ \left\{ v_0^{j_0}, v_1^{j_1}, \dots, v_k^{j_k} \right\} \mid \{v_0, v_1, \dots, v_k\} \in X(k), f(\{v_0, v_1, \dots, v_k\}) = \prod_{i=0}^k j_i \right\}.$$

One can already see that the  $(k-1)$ -skeleton of  $\widehat{X}$  doesn't depend on  $f$  and is just some inflation of the original complex. The dependence on  $f$  is only in the top-level faces. Thus, in particular,  $\widehat{X}$  is not a lift of (the underlying graph) of  $X$ , except when  $k=1$ . However, the links of  $(k-2)$ -faces in  $\widehat{X}$  are lifts of links in  $X$ , which is why we named this complex a local lift. We will see this in the next subsection.

### 3.1 Local Properties of Local Lifts

For the rest of this subsection, we fix  $X$  to be a  $k$ -dimensional pure simplicial complex,  $f : X(k) \rightarrow \{\pm 1\}$  to be a signing function, and  $\widehat{X}$  to be the  $f$ -local lift of  $X$ . We also need the following three pieces of notation:

- Let  $\pi : \widehat{X}(0) \rightarrow X(0)$  be the projection map  $\pi(v^j) = v$ , and we extend it to higher dimensional faces as well by  $\pi(\{v_0^{j_0}, v_1^{j_1}, \dots, v_i^{j_i}\}) = \{v_0, v_1, \dots, v_i\}$ .
- Let  $\text{sign} : \widehat{X} \rightarrow \{\pm 1\}$  be  $\text{sign}(\hat{\sigma}) \rightarrow \prod_{v^j \in \hat{\sigma}} j$ .
- For any  $\hat{\sigma} \in \widehat{X}(k-2)$  with  $\sigma = \pi(\hat{\sigma})$  we denote by  $f_\sigma : X_\sigma(1) \rightarrow \{\pm 1\}$  the function  $f_\sigma(e) = f(\sigma \sqcup e)$  and by  $f_{\hat{\sigma}} : X_{\hat{\sigma}}(1) \rightarrow \{\pm 1\}$  the function  $f_{\hat{\sigma}}(e) = \text{sign}(\hat{\sigma}) \cdot f_\sigma(e)$ .

The first observation is that the degrees of  $\widehat{X}$  are twice the degrees of  $X$ , except for  $d_{k-1}$ , which stays the same.

► **Observation 18.** *If  $X$  is  $(d_0, \dots, d_{k-1})$ -regular then  $\widehat{X}$  is  $(2d_0, 2d_1, \dots, 2d_{k-2}, d_{k-1})$ -regular.* ◄

It is a direct calculation, so we have omitted its proof. We just comment that the reason that  $d_{k-1}$  remains the same is that for every  $\hat{\sigma} \in \widehat{X}(k-1)$  and  $v \in X_{\pi(\hat{\sigma})}(0)$  there is exactly one  $j \in \{\pm 1\}$  such that  $\hat{\sigma} \cup \{v^j\} \in \widehat{X}(k)$ . Therefore,  $d_{k-1}(X) = |X_{\pi(\hat{\sigma})}(0)| = |\widehat{X}_{\hat{\sigma}}(0)| = d_{k-1}(\widehat{X})$ .

The next lemma gives a complete description of the links of  $\widehat{X}$ .

- **Lemma 19** (on the structure of the links). *Let  $\hat{\sigma} \in \widehat{X}$  and denote by  $\sigma = \pi(\hat{\sigma})$ .*
- If  $\dim(\hat{\sigma}) < k-2$ , then the 1-skeleton of  $\widehat{X}_{\hat{\sigma}}$  (the  $\hat{\sigma}$ -link of  $\widehat{X}$ ), is isomorphic to the 1-skeleton of  $X_\sigma$  tensored with the complete graph on two vertices with self loops<sup>3</sup>.*
  - If  $\dim(\hat{\sigma}) = k-2$ , then  $\widehat{X}_{\hat{\sigma}}$  is isomorphic to a lift of  $X_\sigma$  induced by  $f_{\hat{\sigma}}$ .*

**Proof.** The first item directly follows from the definition of a tensor product. For the second item, suppose  $\dim(\hat{\sigma}) = k-2$ . Both the vertices of  $\widehat{X}_{\hat{\sigma}}$  and of the  $f_{\hat{\sigma}}$ -induced lift of  $X_\sigma$  are  $X_\sigma(0) \times \{\pm 1\}$ . As for the edges,  $\{u^i, v^j\} \in \widehat{X}_{\hat{\sigma}}(1)$  if and only if  $\{u, v\} \in X_\sigma(1)$  and  $ij \cdot \text{sign}(\hat{\sigma}) = f(\sigma \sqcup \{u, v\})$  (or equivalently  $ij = f_{\hat{\sigma}}(\{u, v\})$ ). This is precisely the relation that defines edges in the  $f_{\hat{\sigma}}$ -induced lift of  $X_\sigma$ . ◄

<sup>3</sup> Note that this is also true for the link of  $\sigma = \emptyset$ , i.e.  $\widehat{X}$  itself.



The following corollary that bounds the spectrum of the links is direct.

► **Corollary 20.** *Let  $\hat{\sigma} \in \widehat{X}$  and denote  $\sigma = \pi(\hat{\sigma})$ .*

1. *If  $\dim(\hat{\sigma}) < k - 2$  then  $\lambda(\widehat{X}_{\hat{\sigma}}) = \lambda(X_{\sigma})$ .*
2. *If  $\dim(\hat{\sigma}) = k - 2$  then  $\text{Spec}(\widehat{X}_{\hat{\sigma}}) = \text{Spec}(A_{\sigma}) \cup \text{Spec}(A_{\sigma}^{f_{\hat{\sigma}}})$ , where  $A_{\sigma}$  is the normalized adjacency matrices of  $X_{\sigma}$  and  $A_{\sigma}^{f_{\hat{\sigma}}}$  is its signed normalized adjacency matrix with respect to  $f_{\hat{\sigma}}$ .*

**Proof.** The first item follows from the first item in Lemma 19 that shows the link of  $\widehat{X}_{\hat{\sigma}}$  is isomorphic to the link of  $X_{\sigma}$  tensored with a complete graph, and Fact 6 that bounds the expansion of such a graph. The second item follows from Lemma 9 and the fact that the link is the  $f_{\hat{\sigma}}$ -induced lift of  $X_{\sigma}$  as we saw in Lemma 19. ◀

## 4 Families of HDXs via Random Local Lifts

This section is dedicated to existential proofs of high dimensional expanders based on our local lifts from Construction 17. We start by stating the main theorem of this section which asserts that given an arbitrary HDX  $X$ , there exists a family of HDXs with parameters comparable to those of  $X$  so that any member of the family is a local lift of the former. Formally,

► **Theorem 21.** *Let  $X_0$  be a  $(d_0, d_1, \dots, d_{k-1})$ -regular  $\lambda$ -two sided (resp. one sided) HDX over  $n$  vertices, for  $\lambda \in [0, 1]$ . Then there exists a family of  $\max\left\{\lambda, O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right)\right\}$ -two sided (resp. one sided) high dimensional expanders  $\{X_i\}_{i=0}^{\infty}$  so that  $X_i$  is a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular complex over  $2^i n$  vertices and  $X_{i+1}$  is a local lift of  $X_i$ .*

The proof of Theorem 21 is based on proving the single-step version of it, given in Theorem 22, and applying it iteratively.

► **Theorem 22.** *Let  $\lambda \in [0, 1]$ . For any  $k$ -dimensional,  $(d_0, \dots, d_{k-1})$ -regular,  $\lambda$ -two sided (resp. one sided) high dimensional expander  $X$  over  $n$  vertices, there exists a signing  $f : X(k) \rightarrow \{\pm 1\}$  so that  $\widehat{X}$  is a  $\max\left\{\lambda, O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right)\right\}$ -two sided (resp. one sided) high dimensional expander with regularity  $(2d_0, \dots, 2d_{k-2}, d_{k-1})$  and  $2n$  vertices.*

We start by proving Theorem 21 given Theorem 22. The proof of Theorem 22 is more involved and is provided in the remainder of this section.

**Proof of Theorem 21 assuming Theorem 22.** Let  $X_0$  as specified in Theorem 21 and denote  $\lambda' := \max\left\{\lambda, O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right)\right\}$ . The proof is by induction on  $i$ . Clearly  $X_0$  holds the requirements.

For the induction step, let  $X_i$  be a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided (resp. one sided) HDX with  $2^i n$  vertices received in the  $i$ -th step of the process. By Theorem 22, there exists a signing function  $f_i : X_i(k) \rightarrow \{\pm 1\}$  so that the  $f_i$ -local lift of  $X_i$  (denoted by  $\widehat{X}_i$ ) is a  $(2^{i+1} d_0, \dots, 2^{i+1} d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided (resp. one sided) HDX over  $2^{i+1} n$  vertices. Setting  $X_{i+1} := \widehat{X}_i$  concludes the proof. ◀

### 4.1 Proof Outline of Theorem 22

The proof of Theorem 22 is based on Lovász Local Lemma [29] and Lemma 10, and closely follows the lines of the existential proof in [12].

Recall that one approach for proving a given  $k$ -dimensional simplicial complex is  $\lambda$ -HDX, is considering all of its  $\ell$ -links for  $\ell \leq k-2$  and bound the spectrum of each of their 1-skeletons by  $\lambda$ . By Corollary 20, the only links one should be concerned with are those obtained by  $(k-2)$ -faces, as the links of all other faces inherit the expansion from the initial HDX. In addition, by the same corollary, it's enough to analyze the spectra of the signed random walk matrices of the  $(k-2)$ -links of  $X$ , with respect to the signing induced on them as defined in Lemma 19. Indeed, doing so is the most technical part of the proof and follows by the next lemma combined with Lemma 10:

► **Lemma 23.** *For any  $k$ -dimensional pure  $(d_0, \dots, d_{k-1})$ -regular simplicial complex  $X$  over  $n$  vertices, there exists a signing function  $f : X(k) \rightarrow \{\pm 1\}$  such that for any  $(k-2)$ -face  $\hat{\sigma} \in \widehat{X}$  and any disjoint subsets of vertices  $S, T \subseteq X_\sigma(0)$  for  $\sigma = \pi(\hat{\sigma})$ ,*

$$|\langle \mathbf{1}_S, A_\sigma^{f_{\hat{\sigma}}} \mathbf{1}_T \rangle| \leq 10 \sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}} |S||T|} \quad (4.1)$$

where  $f_{\hat{\sigma}}$  is the signing on  $X_\sigma$ 's edges induced by  $f$  as defined in Section 3.1.

By the  $(d_0, \dots, d_{k-1})$ -regularity of  $X$ ,  $X_\sigma$  is a  $d_{k-1}$ -regular graph over  $d_{k-2}$ -vertices. Furthermore, since any signing over the  $k$ -faces induces a signing function on the edges of any  $(k-2)$ -link, our goal is to find a single signing function  $f$  such that these lifts of all the links of the  $(k-2)$ -dimensional faces expand.

The proof of Lemma 23 is by the Lovász Local Lemma Lemma 15. We define the set of “bad” events  $\mathfrak{B} = \{B_\sigma^{S,T}\}$ . The event  $B_\sigma^{S,T}$  is that (4.1) doesn't hold for a fixed  $\sigma \in X(k-2)$  and fixed disjoint sets  $S, T \subseteq X_\sigma(0)$ . In [12], similar bad events were considered, but only the sets  $S, T$  needed to be specified. The main difference between our proof and theirs is that we need to take care of the dependencies between events corresponding to different  $(k-2)$ -faces  $\sigma, \sigma'$ .

To apply the lemma and deduce Lemma 23, one needs to understand and analyze the dependency relation of the events in  $\mathfrak{B}$ .

### On the dependency of bad events in $\mathfrak{B}$

Fix  $\hat{\sigma} \in \widehat{X}(k-2)$  and disjoint  $S, T \subseteq X_\sigma(0)$  for  $\sigma = \pi(\hat{\sigma})$ , and define  $F(\sigma, S, T) \subseteq X(k)$  to be the set of all  $k$ -faces of  $X$  so that  $\sigma \subseteq \tau$  and  $\tau \setminus \sigma$  is an edge in the graph induced by  $S \sqcup T$  on  $X_\sigma$ . Recall that  $\text{sign} : \widehat{X} \rightarrow \{\pm 1\}$  is defined by  $\text{sign}(\hat{\sigma}) = \prod_{v \in \hat{\sigma}} j$ , and note that

$$d_{k-1} \langle \mathbf{1}_S, A_\sigma^{f_{\hat{\sigma}}} \mathbf{1}_T \rangle = \sum_{\substack{uv \in X_\sigma(1) \\ \text{s.t. } u \in S, v \in T}} f_{\hat{\sigma}}(uv) = \text{sign}(\hat{\sigma}) \sum_{\substack{uv \in X_\sigma(1) \\ \text{s.t. } u \in S, v \in T}} f(\sigma \sqcup uv) = \text{sign}(\hat{\sigma}) \sum_{\tau \in F(\sigma, S, T)} f(\tau).$$

Since the signs  $f$  assigns to the  $k$ -faces are chosen independently, if the event  $B_\sigma^{S,T}$  is *not* mutually independent of a subset  $\mathfrak{A} \subseteq \mathfrak{B}$ , there must exist some event  $B_{\sigma'}^{S',T'} \in \mathfrak{A}$  for which  $F(\sigma, S, T) \cap F(\sigma', S', T')$  is not empty.

Towards using the Lovász Local Lemma, we need to bound the probability of the event  $B_\sigma^{S,T}$  as well as the number of neighbors it has in the dependency graph considered in the Local Lemma. The bound on the probability follows directly from the arguments in [12], but bounding the number of neighbors each event is more involved. In contrast to the expander graph case considered in [12] (where the “bad” events only depend on  $S$  and  $T$ ), in simplicial complexes events corresponding to different faces  $\sigma, \sigma'$  (and therefore to different links) can depend on one another as long as they have a common  $k$ -face in  $F(\sigma, S, T) \cap F(\sigma', S', T')$ .

A naive count of the number of such  $k$ -faces won't suffice, and would lead the bound in Equation (4.1) to scale with  $d_{k-2}$ . Therefore, we need to carefully characterize when exactly  $F(\sigma, S, T)$  and  $F(\sigma', S', T')$  intersect.

The first case where dependency can occur is when  $\sigma = \sigma'$ . In this case, we are in the same setting as in [12], which observed that there must be an edge in the subgraph induced by  $S \cup T$  as well as in the one induced by  $S' \cup T'$  for a dependency to happen.

In the second case, which is new to our proof,  $\sigma \neq \sigma'$  meaning that each of the events considers a different graph. In this case, we observe that this implies both that there is a  $k$ -face  $\tau$  containing both  $\sigma, \sigma'$  and that either there exist vertices  $v \in \sigma \cap (S' \sqcup T')$  and  $s \in S \sqcup T$  so that  $\tau \setminus \sigma' = \{v, s\}$ , or that  $\tau \setminus \sigma' \subseteq \sigma$ . As we show below, this characterization is useful to bound the number of possible events that are dependent on a certain  $B_\sigma^{S,T}$ . The following claim gives this characterization formally.

▷ **Claim 24.** Let an event  $B_\sigma^{S,T} \in \mathfrak{B}$  and some subset  $\mathfrak{A} \subseteq \mathfrak{B}$ . If

- for any  $B_{\sigma'}^{S',T'} \in \mathfrak{A}$  with  $\sigma = \sigma'$  there is no edge lying in both  $E_{X_\sigma}(S, T)$  and  $E_{X_{\sigma'}}(S', T')$ , and
  - for any  $B_{\sigma'}^{S',T'} \in \mathfrak{A}$  with  $\sigma \neq \sigma'$  there is no  $k$ -face  $\tau \in X$  containing both  $\sigma$  and  $\sigma'$ , so that  $\tau \setminus \sigma$  and  $\tau \setminus \sigma'$  are edges in  $E_{X_\sigma}(S, T)$  and  $E_{X_{\sigma'}}(S', T')$  respectively,
- then  $B_\sigma^{S,T}$  is mutually independent of  $\mathfrak{A}$ .

All left to conclude the proof of Lemma 23 is carefully counting the number of events that fulfill one of the claim conditions and provide a real function that bounds the probability of each event as in Equation (2.1). We leave the details for the formal proof, which is given in the next section.

## 4.2 Proving Lemma 23

This subsection is dedicated to the proof of Lemma 23 together with the subclaims it requires. We start by setting notations and highlighting features that will be needed for the proof.

### Notations

We say that sets  $S, T \subseteq X_\sigma(0)$  induce a *connected subgraph* if the subgraph obtained by projecting  $X_\sigma$  on  $S \cup T$  is connected. In addition, we write  $E_{X_\sigma}(S, T)$  to indicate the set of edges between  $S$  and  $T$  in  $X_\sigma$ . For  $\hat{\sigma} \in \widehat{X}(k-2)$ , we denote  $\sigma = \pi(\hat{\sigma})$ . When the face  $\hat{\sigma}$  being considered is clear from the context we abbreviate and write  $f$  for  $f_{\hat{\sigma}}$ .

In addition, we rely on the following observation:

► **Observation 25.** *If a signing  $f : X(k) \rightarrow \{\pm 1\}$  independently assigns a uniform sign to each  $k$ -face, then for any  $\hat{\sigma} \in \widehat{X}(k-2)$ ,  $f_{\hat{\sigma}}$  independently assigns a uniform sign to each edge in  $X_\sigma$ .*

**Proof.** Let  $\hat{\sigma} \in \widehat{X}(k-2)$  and let  $e \neq e' \in X_\sigma(1)$ . Then for any  $j, j' \in \{\pm 1\}$

$$\begin{aligned} \mathbb{P}[f_{\hat{\sigma}}(e) = j \wedge f_{\hat{\sigma}}(e') = j'] &= \mathbb{P}[\text{sign}(\hat{\sigma}) \cdot f(\sigma \sqcup e) = j \wedge \text{sign}(\hat{\sigma}) \cdot f(\sigma \sqcup e') = j'] \\ &= \mathbb{P}[\text{sign}(\hat{\sigma}) \cdot f(s \sqcup e) = j] \cdot \mathbb{P}[\text{sign}(\hat{\sigma}) \cdot f(\sigma \sqcup e') = j'] = \frac{1}{4}. \quad \blacktriangleleft \end{aligned}$$

In addition, as in [12], we can restrict the proof to consider only a pair of sets inducing connected subgraphs and deduce the result to any pair of sets. In addition, we can assume that  $d$  is as large as 996 as it is always the case that  $\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle \leq \sqrt{|S||T|}$  and  $1 \leq 10\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}}}$  for  $d_{k-1} \in [1, 996]$ .

We are now ready to prove Lemma 23.

**Proof of Lemma 23.** We set  $f$  to be a randomized signing of  $X(k)$  by setting a uniform and independent sign from  $\{\pm 1\}$  to any  $k$ -face. Fix some face  $\hat{\sigma} \in \widehat{X}(k-2)$  with  $\pi(\hat{\sigma}) = \sigma$ , and disjoint sets  $S, T \subseteq X_\sigma(0)$ . Denote the “bad” event in which the claim does not hold for our fixed face and sets by  $B_\sigma^{S,T}$ . That is,  $\mathbb{P}[B_\sigma^{S,T}] = \mathbb{P}\left[|\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle| > 10\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}} |S||T|}\right]$ .

Fix for a moment some edge  $uv \in X_\sigma(1)$ , and consider the  $(u, v)$ -th entry of  $A_\sigma^f$ . By Definition 8,  $A_\sigma^f(u, v) = \frac{1}{d_{k-1}} f_{\hat{\sigma}}(uv)$ , which, per Observation 25, distributed uniformly in  $\{\pm 1\}$  and is independent of all other edges signs. In addition, since

$$\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle = \frac{1}{d_{k-1}} \sum_{uv \in E_{X_\sigma}(S, T)} f_{\hat{\sigma}}(uv),$$

$\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle$  is a sum of independent uniform random variables over  $\{\pm 1\}$ , implying that

$$\mathbb{E}_f[\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle] = \frac{1}{d_{k-1}} \sum_{uv \in E_{X_\sigma}(S, T)} \mathbb{E}_f[f_{\hat{\sigma}}(uv)] = 0.$$

Hence, by Hoeffding’s inequality,

$$\begin{aligned} \mathbb{P}[B_\sigma^{S,T}] &= \mathbb{P}\left[|\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle| > 10\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}} |S||T|}\right] \\ &\leq 2 \exp\left(-\frac{2 \cdot 100 \frac{k^2 \ln d_{k-1}}{d_{k-1}} |S||T|}{\sum_{uv \in E_{X_\sigma}(S, T)} \left(\frac{1}{d_{k-1}} - \left(-\frac{1}{d_{k-1}}\right)\right)^2}\right). \end{aligned} \quad (4.2)$$

Assuming w.l.o.g. that  $|S| \geq |T|$  we get

$$\begin{aligned} \frac{200 \frac{k^2 \ln d_{k-1}}{d_{k-1}} |S||T|}{\sum_{uv \in E_{X_\sigma}(S, T)} \left(\frac{1}{d_{k-1}} - \left(-\frac{1}{d_{k-1}}\right)\right)^2} &= \frac{200k^2 d_{k-1} (\ln d_{k-1}) |S||T|}{4|E_{X_\sigma}(S, T)|} \geq \frac{50k^2 d_{k-1} (\ln d_{k-1}) |S||T|}{d_{k-1} |T|} \\ &\geq 25k^2 \ln d_{k-1} |S \sqcup T|. \end{aligned}$$

Hence, denoting  $c = |S \sqcup T|$ ,

$$\text{Equation (4.2)} \leq 2 \exp(-25ck^2 \ln d_{k-1}) \leq d_{k-1}^{-10ck^2}. \quad (4.3)$$

We turn to analyze the dependency graph of the “bad” events:<sup>4</sup> Recall that  $\mathfrak{B}$  is the set of all events  $B_\sigma^{S,T}$  for  $\sigma \in X(k-2)$  and disjoint subsets  $S, T \subseteq X_\sigma(0)$ . Using the characterization of correlated events in  $\mathfrak{B}$  given in Claim 24, we get the following bound on the neighborhood size of the events in the dependency graph:

▷ **Claim 26.** Let  $B_\sigma^{S,T} \in \mathfrak{B}$  and denote  $c = |S \sqcup T|$ . Then  $B_\sigma^{S,T}$  has at most  $3k^2 cd^{c'-1}$  neighbors  $B_{\sigma'}^{S',T'}$  with  $|S' \sqcup T'| = c'$ .

Now, to apply Lovász Local Lemma, one needs to define a function  $\rho : \mathfrak{B} \rightarrow [0, 1)$  such that  $\mathbb{P}[B_\sigma^{S,T}] \leq \rho(B_\sigma^{S,T}) \prod_{B_{\sigma'}^{S',T'} \sim B_\sigma^{S,T}} \left(1 - \rho(B_{\sigma'}^{S',T'})\right)$ . Set  $\rho(B_\sigma^{S,T}) = d_{k-1}^{-3ck^2}$ . Indeed

<sup>4</sup> Recall that the dependency graph of a set of events  $\mathfrak{B}$  is a digraph with a vertex for each event  $B \in \mathfrak{B}$  and any event  $B$  is mutually independent of  $\mathfrak{B} \setminus \Gamma(B)$ .

$$\begin{aligned} \rho(B_\sigma^{S,T}) \prod_{B_\sigma^{S',T'} \in \sim B_\sigma^{S,T}} \left(1 - \rho(B_\sigma^{S',T'})\right) &= d_{k-1}^{-3ck^2} \prod_{B_\sigma^{S',T'} \sim B_\sigma^{S,T}} \left(1 - d_{k-1}^{-3|S' \cup T'|k^2}\right) \\ &= d_{k-1}^{-3ck^2} \prod_{c' \in [n]} \left(1 - d_{k-1}^{-3c'k^2}\right)^{2^{c'} 3ck^2 d_{k-1}^{c'-1}} \end{aligned} \quad (4.4)$$

$$\geq d_{k-1}^{-3ck^2} \exp\left(-6ck^2 \sum_{c' \in [n]} 2^{c'} d_{k-1}^{c'-1} d_{k-1}^{-3c'k^2}\right) \quad (4.5)$$

$$\begin{aligned} &\geq d_{k-1}^{-3ck^2} e^{-7ck^2} \\ &\geq d_{k-1}^{-10ck^2} \geq \mathbb{P}[B_\sigma^{S,T}] \end{aligned} \quad (4.6)$$

where Equation (4.4) is since for any  $U \subseteq X_\sigma(0)$  of cardinality  $c'$ , there are at most  $2^{c'}$  pairs of disjoint sets  $S', T'$  with  $S' \sqcup T' = U$ , Equation (4.5) is by  $e^{-x} \leq 1 - \frac{x}{2}$  for any  $x \in [0, 1.59]$  and Equation (4.6) is by taking  $d_{k-1} \geq 3$ . Together with Equation (4.3), this concludes the proof.  $\blacktriangleleft$

The formal proofs of Claim 24 and Claim 26, and the proof of Theorem 22 given Lemma 23 appear in the full version of this paper [11].

### 4.3 Concluding Theorem 22

**Proof of Theorem 22.** Let  $X$  be a  $(d_0, \dots, d_{k-1})$ -regular,  $\lambda(X)$ -two sided (resp. one sided) HDX over  $n$  vertices, fix  $f$  to be the signing provided by Lemma 23, and set  $\widehat{X}$  to be the  $f$ -local lift of  $X$ .

By Observation 18,  $\widehat{X}$  is a  $(2d_0, \dots, 2d_{k-2}, d_{k-1})$ -regular graph over  $2n$  vertices. We need to prove that for any  $\widehat{\sigma} \in \widehat{X}$  of dimension  $\leq k-2$ , the 1-skeleton of  $\widehat{X}_{\widehat{\sigma}}$  is a  $\max\left\{\lambda(X), O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right)\right\}$ -two sided (resp. one sided) expander.

By Corollary 20, the spectra of all links  $\widehat{X}_{\widehat{\sigma}}$  with  $\widehat{\sigma}$  of dimension  $< k-2$  are bounded by  $\lambda(X)$ . In addition, by Lemma 23, for any  $\widehat{\sigma} \in \widehat{X}(k-2)$  and any disjoint sets  $S, T \subseteq X_\sigma(0)$  for  $\sigma = \pi(\widehat{\sigma})$ , we have that  $|\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle| \leq O\left(\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}} |S||T|}\right)$  where  $A_\sigma^f$  is the  $f_\sigma$ -signed random walk matrix of  $X_\sigma$ . Together with Lemma 10 this implies

$$\begin{aligned} \lambda(X_\sigma) &= O\left(\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}} \left(1 + \log\left(\sqrt{\frac{d_{k-1}}{k^2 \log d_{k-1}}}\right)\right)}\right) \\ &\leq O\left(\sqrt{\frac{k^2 \log d_{k-1}}{d_{k-1}}} \cdot \log \sqrt{d_{k-1}}\right) = O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right) \end{aligned}$$

hence, by Lemma 9,  $\lambda(X_\sigma) = \max\left\{\lambda(X), O\left(\sqrt{\frac{k^2 \log^3 d_{k-1}}{d_{k-1}}}\right)\right\}$  and by Corollary 20, this is also the case for  $\lambda(\widehat{X}_{\widehat{\sigma}})$ .  $\blacktriangleleft$

## 5 An Algorithmic Version of Theorem 21

In this subsection, we prove that there is a randomized algorithm that finds a family of local lifts as in Theorem 21 when  $X$  is a high dimensional expander under mild assumptions on the degree sequence which we encapsulate in the following definition.

► **Definition 27** (Nice complex). *Let  $X$  be a  $k$ -dimensional simplicial complex. We say that  $X$  is nice if  $X$  is regular, and*

$$d_{k-2}^{1-4 \log d_{k-1}} < \frac{2}{e(k+1)kd_{k-1} + 1}. \quad (5.1)$$

We prove the following.

► **Theorem 28.** *There exists a randomized algorithm  $\mathcal{A}$  that takes as input a  $k$ -dimensional complex  $X_0$  and an integer  $i \geq 1$ , runs in expected time  $\text{poly}((2^i |X_0(0)|)^k)$  at most, and outputs a  $k$ -dimensional complex  $X_i$  with  $2^i |X_0(0)|$  vertices. The algorithm has the following guarantee.*

*If  $X_0$  is a nice  $(d_0, \dots, d_{k-1})$ -regular  $\lambda$ -two sided high dimensional expander, then  $X_i$  is a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided high dimensional expander where  $\lambda' = O\left(\max\left\{\lambda\left(1 + \log \frac{1}{\lambda}\right), \sqrt{\frac{\log^3 d_{k-1}}{d_{k-1}}}\right\}\right)^5$ .*

*Moreover, one can modify the algorithm so that it outputs a sequence  $X_1, X_2, \dots, X_i$  of complexes all satisfying the same guarantees (instead of just the last one), so that for every  $j = 0, 1, \dots, i-1$ ,  $X_{j+1}$  is a local lift of  $X_j$ .*

Loosely speaking, in order to prove Theorem 28, we need to prove that there is an algorithm  $\mathcal{A}$  that finds a single local lift for  $X$  in polynomial time (just as Theorem 21 was proved by the “one-step theorem” Theorem 22) with good enough spectral expansion. Then we just iteratively use  $\mathcal{A}$  on its own output, setting  $X_{j+1} = \mathcal{A}(X_j)$ , until reaching  $j = i-1$ . For this to work, we also need to address the issue that  $\lambda' \geq \lambda$  so naively the expansion deteriorates as we reiterate. We defer the proof of Theorem 28 for the full version of this paper [11].

► **Remark 29** (A non-uniform algorithm for any HDX). Theorem 28 requires that  $X_0$  be a nice complex, i.e. that (5.1) holds. However, in any family  $\{X_i\}_{i=0}^\infty$  where  $X_{i+1}$  is a local lift of  $X_i$ , the degree  $d_{k-2}(X_i)$  tends to infinity with  $i$  while the other side of both inequalities stays fixed. Thus, the inequalities will eventually hold for any family of consecutive local lifts. In fact, they should hold for any  $i \geq C \log(k + d_{k-1}(X_0))$  for some large enough constant  $C > 0$ . Thus we can modify the algorithm to work even if  $X_0$  is not nice (albeit with the spectral expansion bound guaranteed in Theorem 21, which is slightly worse than the expansion in Theorem 28). This is done by allowing the algorithm to do a brute-force search for the first few steps, to produce a nice  $X_i$ , and then continuing as the original algorithm does. The first few steps will eventually stop because Theorem 21 promises the existence of such an  $X_j$ . This process takes  $\text{poly}(|X_i|^k) + \exp(O(|X_0|^k))$  time.

Towards the proof of Theorem 28, we need the following definition and lemma from [12].

► **Definition 30.** *A graph  $G$  with adjacency operator  $A$  is said to be  $(\beta, t)$ -sparse if for every  $S, T \subseteq V(G)$  such that  $|S \cup T| \leq t$ ,  $\langle \mathbf{1}_S, A \mathbf{1}_T \rangle \leq \beta \sqrt{|S||T|}$ . For a  $k$ -dimensional hyper-regular complex  $X$ , we say that it is  $(\beta, t)$ -sparse if for every  $\sigma \in X(k-2)$ , the graph  $X_\sigma$  is  $(\beta, t)$ -sparse.*

► **Remark 31.** While the definition here regards any  $S, T$  with  $|S \cup T| \leq t$ , it is in fact equivalent to regarding only  $S, T$  with  $|S \cup T| \leq t$  such that the graph induced on  $S \cup T$  is connected. We also remark that if  $X$  is  $(\beta, t)$ -sparse then it is also  $(\beta', t)$ -sparse for any  $\beta' \geq \beta$ .

<sup>5</sup> We will not calculate the constants in the big  $O$  notation explicitly.

The reason we need this definition of sparseness is that in a random local lift, sparseness does not deteriorate with high probability. More formally, the following lemma was proven in [12].

► **Lemma 32** ([12, Lemma 3.4]). *Let  $G = (V, E)$  be a  $d$ -regular graph with  $n$  vertices that is  $(\beta, \log n)$ -sparse for  $\beta \geq 10\sqrt{\frac{\log d}{d}}$ . Then with probability  $\geq 1 - n^{-4 \log d}$  over  $f : E \rightarrow \{\pm 1\}$ :*

- For every  $S, T \subseteq V$ ,  $|\langle \mathbf{1}_S, A^f \mathbf{1}_T \rangle| \leq \beta \sqrt{|S||T|}$  and,
- $\widehat{G}^f$  is  $(\beta, \log n + 1)$ -sparse,

We comment that [12, Lemma 3.4] does not explicitly calculate the probability  $1 - n^{-4 \log d}$ ; rather, they only say the events happen with high probability. This is the probability that is implicit in their proof. They also prove this theorem for  $\beta = 10\sqrt{\frac{\log d}{d}}$  but the same proof extends to  $\beta \geq 10\sqrt{\frac{\log d}{d}}$  with no additional changes.

This next claim easily follows from the definition of expansion and says that a spectral expander is sparse.

▷ **Claim 33.** Let  $G$  be a  $d$ -regular  $\lambda$ -two sided spectral expander over  $n$  vertices such that  $\lambda > \frac{1}{\sqrt{d}}$  and  $d \geq 3$ . Then  $G$  is  $(2\lambda, \log n)$ -sparse.

Proof. Fix  $S, T$  such that with  $|S \cup T| \leq \log n$ . By the  $\lambda$ -expansion and the expander mixing lemma (see e.g. [40]),  $\langle \mathbf{1}_S, A \mathbf{1}_T \rangle \leq \frac{|S||T|}{n} + \lambda \sqrt{|S||T|}$ . We bound this term by  $(\lambda + \frac{\log n}{n}) \sqrt{|S||T|}$ . As  $\frac{\log n}{n} \leq \frac{1}{\sqrt{n}} \leq \frac{1}{\sqrt{d}} \leq \lambda$  the claim follows. ◁

We are ready to state our one-step theorem.

► **Theorem 34.** *There exists a randomized algorithm  $\mathcal{A}$  with the following guarantees. Let  $X$  be a  $k$ -dimensional  $\bar{d}$ -regular  $\lambda$ -two sided (resp. one sided) high dimensional expander over  $n$  vertices, where  $\bar{d} = (d_0, \dots, d_{k-1})$ . Let  $\beta \geq 10\sqrt{\frac{\log d_{k-1}}{d_{k-1}}}$  and denote by*

$\lambda' = \max \left\{ \lambda, O(\beta(1 + \log \frac{1}{\beta})) \right\}$ . *Assume that  $X$  is  $(\beta, \log d_{k-2})$ -sparse, and suppose that  $d_{k-2}$  and  $d_{k-1}$  satisfy  $d_{k-2} > d_{k-1}^2$  and (5.1). Then  $\mathcal{A}(X) = \widehat{X}$  is a local lift of  $X$  such that:*

1. *The complex  $\widehat{X}$  is a  $\lambda'$ -two sided (resp. one sided) high dimensional expander.*
2. *The complex  $\widehat{X}$  is  $(\beta, \log 2d_{k-2})$ -sparse.*

*Upon input  $X$  satisfying the above, the algorithm runs in time  $\text{poly}(|X(0)|^k)$ .*

**Proof of Theorem 34.** We intend to use Lemma 16. For this, we fix the following “bad” events  $\mathfrak{C} = \{C_\sigma \mid \sigma \in X(k-2)\}$  where  $C_\sigma \subseteq \{f : X(k) \rightarrow \{\pm 1\}\}$  is the event where :

1. Either  $\widehat{X}_\sigma^{\pm f_\sigma}$  is not a  $\lambda'$ -two sided spectral expander, or
2.  $\widehat{X}_\sigma^{\pm f_\sigma}$  is not  $(\beta, \log 2d_{k-2})$ -sparse.

By Lemma 32 (and Lemma 10 that relates the first item in Lemma 32 to spectral expansion),  $\mathbb{P}_f [C_\sigma] \leq 2d_{k-2}^{-4 \log d_{k-1}}$ . Moreover, because every link of a  $\hat{\sigma} \in \widehat{X}(k-2)$  is a lift of  $X_\sigma$  with respect to  $f_{\hat{\sigma}}$ , then if none of the events  $C_\sigma$  occur, then  $\widehat{X}$  satisfies both items in Theorem 34. We will use Lemma 16 to find such an assignment.

We now construct a dependency graph for  $\mathfrak{C}$ . Let  $\sigma \in X(k-2)$  and  $U \subseteq X(k-2)$ . The event  $C_\sigma$  only depends on  $f_\sigma$ , so it only depends on  $k$ -faces  $\tau \supseteq \sigma$ . Therefore, if  $C_\sigma$  and  $\{C_{\sigma'} \mid \sigma' \in U\}$  are not mutually independent, then in particular there is a  $k$ -face  $\tau \in X$  and  $\sigma' \in U$  such that  $\tau \supseteq \sigma, \sigma'$ . Hence, in our dependency graph we connect  $C_\sigma \sim C_{\sigma'}$  if there exists such a  $k$ -face containing both  $\sigma$  and  $\sigma'$ . Let us upper bound the neighborhood size of an event  $C_\sigma$ . The number of neighbors that  $C_\sigma$  has is upper bounded by the number of  $k$ -faces containing  $\sigma$  times  $\binom{k+1}{k-1}$  (the number of ways to choose  $\sigma' \subseteq \tau$ ). Therefore, the number of neighbors is bounded by

$$D := \binom{k+1}{k-1} \cdot |\{\tau \in X(k) \mid \tau \supseteq \sigma\}| = \binom{k+1}{k-1} |X_\sigma(1)| = \frac{(k+1)k}{4} d_{k-2} d_{k-1}.$$

By setting  $\rho : \mathfrak{C} \rightarrow [0, 1)$  to be the constant function  $\rho(C_\sigma) = \frac{1}{D+1}$  we have that  $\mathbb{P}[C_\sigma] \leq \rho(C_\sigma) \prod_{\sigma' \sim \sigma} (1 - \rho(C_{\sigma'}))$ , because  $\rho(C_\sigma) \prod_{\sigma' \sim \sigma} (1 - \rho(C_{\sigma'})) \geq \frac{1}{D+1} \left(1 - \frac{1}{D+1}\right)^D \geq \frac{1}{e(D+1)}$  and  $\mathbb{P}[C_\sigma] \leq 2d_{k-2}^{-4 \log d_{k-1}} \leq \frac{1}{e(D+1)}$  by (5.1).

Let us now verify that the algorithm in Lemma 16 runs in polynomial time. We note that there the number of events in  $\mathfrak{C}$  is  $\text{poly}(|X(0)|^k)$ , and checking whether  $C_\sigma$  occurs could be done in  $\text{poly}(|X(0)|)$ -time because it amounts to:

1. Find the spectrum of a signed adjacency operator of a  $d_{k-2}$ -sized graph.
  2. Going over all connected sets  $U \subseteq \widehat{X}_\sigma^{\pm f_\sigma}$  of size  $\leq \log 2d_{k-2}$  for every  $\sigma \in X(k-2)$ , finding  $S, T$  such that  $S \cup T = U$ , and counting the number of edges between  $S$  and  $T$  to check if the pair  $S, T$  violates sparseness. There is a  $\text{poly}(|X(0)|)$  such  $U, S, T$  at most.
- Therefore, the randomized algorithm in Lemma 16 will find a signing in  $\text{poly}(|X(0)|^k) \cdot \sum_{\sigma \in X(k-2)} \frac{1}{D} = \text{poly}(|X(0)|^k)$ -time.  $\blacktriangleleft$

## 6 Derandomizing the Construction

In this section we provide a deterministic construction of  $(k-1)$ -bounded families of high dimensional expanders, as referred to in Theorem 3. For the rest of this section, we denote  $\alpha_k(d) = 10\sqrt{\frac{k^2 \log d}{d}}$  (when  $k$  is clear from context, we will write  $\alpha(d)$ ).

We will prove the following theorem.

**► Theorem 35** (Restatement of Theorem 3). *There exists a deterministic algorithm  $\mathcal{B}$  that takes as input a  $k$ -dimensional complex  $X_0$  and an integer  $i \geq 1$ , runs in time  $\text{poly}((2^i |X_0(0)|)^k)$ , and outputs a  $k$ -dimensional complex  $X_i$  with  $2^i |X_0(0)|$  vertices. The algorithm has the following guarantee: If  $X_0$  is a  $(d_0, \dots, d_{k-1})$ -regular  $\lambda$ -two sided high dimensional expander, with  $\lambda > \alpha_k(d_{k-1})$ ,  $d_{k-1} > 2^{10k}$  and  $|X_0(k-2)| \leq d_{k-2}^{10k}$ , then  $X_i$  is a  $(2^i d_0, \dots, 2^i d_{k-2}, d_{k-1})$ -regular  $\lambda'$ -two sided high dimensional expander where  $\lambda' = O(2^{5k} \lambda (1 + \log \frac{1}{\lambda}))^6$ . In particular, for every  $n \in \mathbb{N}$ , choosing  $i = \log n$  yields a complex with at least  $n$ -vertices.*

This explicit construction generalizes the explicit construction for expanders given in [12], which is based on the conditional probabilities method [4, Chapter 16].

We first observe that under the assumption that the base complex is sparse (as in Definition 30) and that  $|X(k-2)|$  is not too large, then a random local lift of  $X$  is also sparse and is a high dimensional expander with high probability. Then, we explain how we can find such a lift deterministically by greedily selecting the values  $f(\tau)$  one  $k$ -face at a time.

**► Lemma 36.** *Let  $X$  be a  $k$ -dimensional,  $(d_0, \dots, d_{k-1})$ -regular and  $(\beta, \log d_{k-2})$ -sparse simplicial complex so that  $\beta \geq \alpha(d_{k-1})$  and  $|X(k-2)| \leq d_{k-2}^{\log d_{k-1}}$ .*

*Then, for  $f : X(k) \rightarrow \{\pm 1\}$  drawn uniformly at random, with probability at least  $1 - d_{k-2}^{-3 \log d_{k-1}}$ :*

1. *For every  $\sigma \in X(k-2)$  and every  $S, T \subseteq X_\sigma(0)$ :  $|\langle \mathbf{1}_S, A_\sigma^f \mathbf{1}_T \rangle| \leq \beta \sqrt{|S||T|}$ .*
2. *The local lift  $\widehat{X} = \widehat{X}^f$  is  $(\beta, \log d_{k-2} + 1)$ -sparse.*  $\blacktriangleright$

<sup>6</sup> We will not calculate the constants in the big  $O$  notation explicitly.



We comment that the condition  $|X(k-2)| \leq d_{k-2}^{\log d_{k-1}}$  may seem odd at first glance. However, similar to Remark 29, this is eventually satisfied by every sequence  $\{X_i\}_{i=0}^\infty$  where  $X_{i+1}$  is a local lift of  $X_i$ . Thus, we do not lose too much generality by assuming it.

The proof of Lemma 36 follows by applying Lemma 32 to every link and taking a union bound over the links. We omit the proof since it is a direct calculation.

The deterministic construction mentioned at the beginning of this section is composed of iterative applications of the local lift, where each application is according to the algorithm described in the following lemma.

► **Lemma 37.** *Let  $X$  be a  $k$ -dimensional  $(d_0, \dots, d_{k-1})$ -regular  $(\beta, \log d_{k-2})$ -sparse simplicial complex with  $d_{k-1} > 2^{10k}$ ,  $\beta \geq \alpha(d_{k-1})$  and such that  $|X(k-2)| \leq d_{k-2}^{10k}$ .*

*Then, there is a deterministic  $\text{poly}(|X(0)|^k)$  time algorithm for finding a function  $f : X(k) \rightarrow \{\pm 1\}$  such that:*

1. *For every  $\sigma \in X(k-2)$ ,  $\|A_\sigma^f\| = O\left(2^{5k}\beta\left(1 + \log \frac{1}{\beta}\right)\right)$ .*
2.  *$\widehat{X}^f$  is  $(\beta, \log d_{k-2} + 1)$ -sparse.*

The proof of Lemma 37 appears in the full version of this paper [11]. We give here a short discussion of the techniques used there. The proof uses the method of conditional probabilities. The main idea is that, given the conditions on the input complex, we can define random variables denoted  $Z^{(\sigma)}$ , which serve as “error” indicators, where these errors occur with very small probability. By defining another set of random variables  $Y^{(\sigma)}$  which correlate with the links’ expansions, and amplifying the impact of each error, we are able to choose  $f(\tau)$   $k$ -face by  $k$ -face, while tracking the expected value of the sum of those variables efficiently and making sure no error occurs. We are now ready to prove our main result in this section.

**Proof of Theorem 35.** Let  $\bar{d} = (d_0, d_1, \dots, d_{k-1} = d)$  and let  $X_0$  be a  $\bar{d}$ -regular  $\lambda$ -two sided high dimensional expander for  $\lambda > \alpha_k(d)$ , such that  $|X_0(k-2)| \leq d_{k-2}^{10k}$ . By Claim 33, it is also  $(2\lambda, \log d_{k-2})$ -sparse.

Denote by  $\mathcal{B}'$  the algorithm suggested by Lemma 37, and let  $X_1, X_2, \dots, X_i$  be such that  $X_j = \mathcal{B}'(X_{j-1})$  for  $j \in [i]$ . We set  $X_i$  to be  $\mathcal{B}'$ ’s output.

Let us show that,  $X_i$  meets the guarantees of Theorem 35. By Observation 18, for every  $j \in [i]$ ,  $X_j$  is  $(2^j d_0, 2^j d_1, \dots, 2^j d_{k-2}, d)$ -regular and  $|X_j(0)| = 2^j |X_0(0)|$ .

In addition, one can verify by a direct calculation that, for any  $j \in [i]$ ,  $|X_j(k-2)| = 2^{k-1} |X_{j-1}(k-2)|$ , so if  $|X_{j-1}(k-2)| \leq d_{k-2} (X_{j-1})^{10k}$  then  $|X_j(k-2)| = 2^{k-1} |X_{j-1}(k-2)| \leq d_{k-2} (X_{j-1})^{10k} \cdot 2^{k-1} \leq d_{k-2} (X_j)^{10k}$ . Thus, by induction and the fact that this inequality holds for  $X_0$ , this holds for every  $j$ .

Finally, by Lemma 37 one inductively obtains that for any  $j$ :

1.  $X_j$  is an  $O\left(2^{5k}\lambda\left(1 + \log \frac{1}{\lambda}\right)\right)$ -high dimensional expander.
2.  $X_j$  is  $(2\lambda, \log d_{k-2}(X_j))$ -sparse.
3.  $X_j$  computed in time  $\text{poly}(|X_{j-1}(0)|^k) = \text{poly}(2^{j-1}|X_0(0)|^k)$ .

as required. ◀

---

## References




- 1 Louigi Addario-Berry and Simon Griffiths. The spectrum of random lifts. *arXiv preprint*, 2010. [arXiv:1012.4097](#).
- 2 Naman Agarwal, Alexandra Kolla, and Vivek Madan. Small lifts of expander graphs are expanding. *CoRR*, abs/1311.3268, 2013. [arXiv:1311.3268](#).

- 3 Vedat Levi Alev and Ori Parzanchevski. Sequential sweeps and high dimensional expansion. *arXiv preprint*, 2023. [arXiv:2312.02089](https://arxiv.org/abs/2312.02089).
- 4 Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.
- 5 Alon Amit and Nathan Linial. Random graph coverings i: General theory and graph connectivity. *Combinatorica*, 22(1):1–18, 2002.
- 6 Alon Amit, Nathan Linial, Jiří Matoušek, and Eyal Rozenman. Random lifts of graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 883–894. Citeseer, 2001. doi:10.1145/365411.365801.
- 7 Anurag Anshu, Nikolas P Breuckmann, and Chinmay Nirkhe. Nlts hamiltonians from good quantum codes. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1090–1096, 2023.
- 8 Mitali Bafna, Noam Lifshitz, and Dor Minzer. Constant degree direct product testers with small soundness, 2024. [arXiv:2402.00850](https://arxiv.org/abs/2402.00850).
- 9 Mitali Bafna and Dor Minzer. Characterizing direct product testing via coboundary expansion. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1978–1989, 2024.
- 10 Cristina M Ballantine. Ramanujan type buildings. *Canadian Journal of Mathematics*, 52(6):1121–1148, 2000.
- 11 Inbar Ben Yaacov, Yotam Dikstein, and Gal Maor. Sparse high dimensional expanders via local lifts, 2024. [arXiv:2405.19191](https://arxiv.org/abs/2405.19191).
- 12 Yehonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica*, 26(1439-6912):495–519, 2006. doi:10.1007/s00493-006-0029-7.
- 13 Aart Blokhuis and Andries E Brouwer. Locally 4-by-4 grid graphs. *Journal of graph theory*, 13(2):229–244, 1989.
- 14 Charles Bordenave. A new proof of Friedman’s second eigenvalue theorem and its extension to random lifts. *Ann. Sci. Éc. Norm. Supér. (4)*, 53(6):1393–1439, 2020. doi:10.24033/asens.2450.
- 15 Morton Brown and Robert Connelly. On graphs with a constant link, ii. *Discrete Mathematics*, 11(3):199–232, 1975.
- 16 Donald I Cartwright, Patrick Solé, and Andrzej Żuk. Ramanujan geometries of type an. *Discrete mathematics*, 269(1-3):35–43, 2003.
- 17 Michael Chapman, Nathan Linial, and Yuval Peled. Expander graphs – both local and global. *Combinatorica*, 40:473–509, 2020. doi:10.1007/s00493-019-4127-8.
- 18 Michael Chapman and Alexander Lubotzky. Stability of homomorphisms, coverings and cocycles ii: Examples, applications and open problems. *arXiv preprint*, 2023. [arXiv:2311.06706](https://arxiv.org/abs/2311.06706).
- 19 Itay Cohen, Roy Roth, and Amnon Ta-Shma. Hdx condensers. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1649–1664. IEEE, 2023.
- 20 David Conlon. Hypergraph expanders from cayley graphs. *Israel Journal of Mathematics*, 233(1):49–65, 2019.
- 21 David Conlon, Jonathan Tidor, and Yufei Zhao. Hypergraph expanders of all uniformities from cayley graphs. *Proceedings of the London Mathematical Society*, 121(5):1311–1336, 2020.
- 22 Yotam Dikstein. New high dimensional expanders from covers. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 826–838, 2023.
- 23 Yotam Dikstein and Irit Dinur. Agreement theorems for high dimensional expanders in the small soundness regime: the role of covers, 2023. [arXiv:2308.09582](https://arxiv.org/abs/2308.09582).
- 24 Yotam Dikstein, Irit Dinur, and Alexander Lubotzky. Low acceptance agreement tests via bounded-degree symplectic hdxs, 2024. URL: <https://eccc.weizmann.ac.il/report/2024/019/>.
- 25 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.



- 26 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 357–374, 2022.
- 27 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *Proc. 58th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 974–985, 2017. doi:10.1109/FOCS.2017.94.
- 28 Irit Dinur, Siqi Liu, and Rachel Yun Zhang. New codes on high dimensional expanders, 2023. arXiv:2308.15563.
- 29 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions, in “infinite and finite sets”(a. hajnal et al., eds.). In *Colloq. Math. Soc. J. Bolyai*, volume 11, page 609, 1975.
- 30 Jacob Fox, Mikhail Gromov, Vincent Lafforgue, Assaf Naor, and János Pach. Overlap properties of geometric expanders. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 2012(671):49–83, 2012.
- 31 Ehud Friedgut and Yonatan Iluz. Hyper-regular graphs and high dimensional expanders, 2020. arXiv:2010.03829.
- 32 Joel Friedman. *A proof of Alon’s second eigenvalue conjecture and related problems*. American Mathematical Soc., 2008.
- 33 Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation: In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 302–332. Springer, 2011.
- 34 Louis Golowich. Improved Product-Based High-Dimensional Expanders. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.38.
- 35 Louis Golowich. From grassmannian to simplicial high-dimensional expanders. *arXiv preprint*, 2023. arXiv:2305.02512.
- 36 Roy Gotlib and Tali Kaufman. List agreement expansion from coboundary expansion, 2022. arXiv:2210.15714.
- 37 Roy Gotlib and Tali Kaufman. No where to go but high: A perspective on high dimensional expanders. *arXiv e-prints*, pages arXiv-2304, 2023.
- 38 M. Gromov. Singularities, expanders and topology of maps. part 2: from combinatorics to topology via algebraic isoperimetry. *Geom. Funct. Anal.*, 20:416–526, 2010. doi:10.1007/s00039-010-0073-8.
- 39 Prahladh Harsha and Ramprasad Saptharishi. A note on the elementary construction of high-dimensional expanders of kaufman and oppenheim, 2019. arXiv:1912.11225.
- 40 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.
- 41 Tali Kaufman and Alexander Lubotzky. High dimensional expanders and property testing. In *Innovations in Theoretical Computer Science, ITCS’14, Princeton, NJ, USA, January 12-14, 2014*, pages 501–506, 2014.
- 42 Tali Kaufman and Izhar Oppenheim. Construction of new local spectral high dimensional expanders. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 773–786, 2018.
- 43 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Comb.*, 40(2):245–281, 2020. doi:10.1007/s00493-019-3847-0.
- 44 Peter Keevash. The existence of designs. *arXiv preprint*, 2014. arXiv:1401.3665.
- 45 Wen-Ching Winnie Li. Ramanujan hypergraphs. *Geometric & Functional Analysis GFA*, 14:380–399, 2004.

- 46 Nathan Linial and Roy Meshulam. Homological connectivity of random 2-complexes. *Combinatorica*, 26:475–487, 2006. doi:10.1007/s00493-006-0027-9.
- 47 Siqi Liu, Sidhanth Mohanty, Tselil Schramm, and Elizabeth Yang. Local and global expansion in random geometric graphs. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 817–825, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3564246.3585106.
- 48 Siqi Liu, Sidhanth Mohanty, and Elizabeth Yang. High-Dimensional Expanders from Expanders. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, 2020. doi:10.4230/LIPIcs.ITCS.2020.12.
- 49 Alexander Lubotzky. High dimensional expanders. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 705–730. World Scientific, 2018.
- 50 Alexander Lubotzky, Zur Luria, and Ron Rosenthal. Random steiner systems and bounded degree coboundary expanders of every dimension. *Discrete & Computational Geometry*, 62:813–831, 2019.
- 51 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- 52 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type  $\tilde{A}_d$ . *European J. Combin.*, 26(6):965–993, 2005. doi:10.1016/j.ejc.2004.06.007.
- 53 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of type  $\tilde{A}_d$ . *Israel J. Math.*, 149(1):267–299, 2005. doi:10.1007/BF02772543.
- 54 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families i: Bipartite ramanujan graphs of all degrees. *Annals of Mathematics*, 182:307–325, 2015. doi:10.4007/annals.2015.182.1.7.
- 55 Gregoty Margulis. Explicit constructions of expanders. *Problemy Peredaci Informacii*, 9(4):71–80, 1973.
- 56 Roy Meshulam and N. Wallach. Homological connectivity of random  $k$ -dimensional complexes. *Random Struct. Algorithms*, 34(3):408–417, 2009. doi:10.1002/rsa.20238.
- 57 Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010.
- 58 A. Márquez, A. de Mier, M. Noy, and M.P. Revuelta. Locally grid graphs: classification and tutte uniqueness. *Discrete Mathematics*, 266(1):327–352, 2003. The 18th British Combinatorial Conference. doi:10.1016/S0012-365X(02)00818-X.
- 59 Ryan O’Donnell and Kevin Pratt. High-dimensional expanders from chevalley groups. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 18:1–18:26. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.18.
- 60 Roberto Imbuzeiro Oliveira. The spectrum of random  $k$ -lifts of large graphs (with possibly large  $k$ ). *arXiv preprint*, 2009. arXiv:0911.4741.
- 61 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. 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 375–388. ACM, 2022. doi:10.1145/3519935.3520017.
- 62 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):1–24, 2008.
- 63 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of mathematics*, 155(1):157–187, 2002.
- 64 Michael Sipser and Daniel A Spielman. Expander codes. *IEEE transactions on Information Theory*, 42(6):1710–1722, 1996.
- 65 David Surowski. Covers of simplicial complexes and applications to geometry. *Geometriae Dedicata*, 16:35–62, April 1984. doi:10.1007/BF00147420.
- 66 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, 2017.

# Randomness Extractors in $AC^0$ and $NC^1$ : Optimal up to Constant Factors

Kuan Cheng   

CFCS, School of CS, Peking University, China

Ruiyang Wu  

CFCS, School of CS, Peking University, China

---

## Abstract

We study randomness extractors in  $AC^0$  and  $NC^1$ . For the  $AC^0$  setting, we give a logspace-uniform construction such that for every  $k \geq n/\text{poly log } n$ ,  $\varepsilon \geq 2^{-\text{poly log } n}$ , it can extract from an arbitrary  $(n, k)$  source, with a small constant fraction entropy loss, and the seed length is  $O(\log \frac{n}{\varepsilon})$ . The seed length and output length are optimal up to constant factors matching the parameters of the best polynomial time construction such as [13]. The range of  $k$  and  $\varepsilon$  almost meets the lower bound in [10] and [7]. We also generalize the main lower bound of [10] for extractors in  $AC^0$ , showing that when  $k < n/\text{poly log } n$ , even strong dispersers do not exist in non-uniform  $AC^0$ . For the  $NC^1$  setting, we also give a logspace-uniform extractor construction with seed length  $O(\log \frac{n}{\varepsilon})$  and a small constant fraction entropy loss in the output. It works for every  $k \geq O(\log^2 n)$ ,  $\varepsilon \geq 2^{-O(\sqrt{k})}$ .

Our main techniques include a new error reduction process and a new output stretch process, based on low-depth circuit implementations for mergers, condensers, and somewhere extractors.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Expander graphs and randomness extractors; Theory of computation  $\rightarrow$  Pseudorandomness and derandomization

**Keywords and phrases** randomness extractor, uniform  $AC^0$ , error reduction, uniform  $NC^1$ , disperser

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.69

**Category** RANDOM

**Related Version** Full Version: <https://ecc.weizmann.ac.il/report/2024/040/>

## 1 Introduction

Randomness extractors are functions that can transform weak random sources into distributions close to uniform. A typical definition of weak random sources is by min-entropy. A random variable (weak source)  $X$  has min-entropy  $k$  if for every  $x$  in the support of  $X$ ,  $\log \frac{1}{\Pr[X=x]} \geq k$ . To extract from an arbitrary weak source of a certain min-entropy, Nisan and Zuckerman [23] introduced the definition of seeded extractor, where the extractor has a short uniform random seed as an extra input. Specifically, a function  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is defined to be a strong  $(k, \varepsilon)$ -extractor, if for every source  $X$  with min-entropy  $k$ ,

$$\|(U_d, \text{EXT}(X, U_d)) - U_{d+m}\| \leq \varepsilon,$$

where  $U_d$  and  $U_m$  are uniform distributions over  $\{0, 1\}^d$  and  $\{0, 1\}^m$  respectively, and  $\|\cdot\|$  is the statistical distance. The entropy loss of such a strong extractor is  $k - m$ . On the contrary, a weak  $(k, \varepsilon)$ -extractor has the same definition except we only require

$$\|\text{EXT}(X, U_d) - U_m\| \leq \varepsilon.$$

The entropy loss of such a weak extractor is  $k + d - m$ .



© Kuan Cheng and Ruiyang Wu;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 69; pp. 69:1–69:22



Leibniz International Proceedings in Informatics

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

As a fundamental pseudorandom construction, extractors are closely related to other pseudorandom objects and also have various applications in computational complexity, combinatorics, algorithm design, information theory, and cryptography. See surveys [21, 29, 37, 30, 1, 38].

Optimizing extractor constructions aims to get, for every  $k$  and  $\varepsilon$ , an extractor with  $d$  as small as possible, and  $m$  as large as possible. An existential bound for strong extractors can be given by a probabilistic argument, which has  $d = \log(n - k) + 2\log(1/\varepsilon) + O(1)$ ,  $m = k - 2\log(1/\varepsilon) - O(1)$ . This is optimal up to some additive constants for  $k \leq n/2$ , due to the lower bound by [24]. After [23], a long line of work has been done to seek explicit extractors with parameters close to the existential bounds [40, 31, 11, 32, 41, 25, 21, 27, 36, 33, 26, 20, 13, 34, 9, 19]. Among them, [13] first achieves  $d = \log n + O(\log(k/\varepsilon))$  and an arbitrary constant factor entropy loss, and also achieves  $m = k - 2\log(1/\varepsilon) - O(1)$  with  $d = \log n + O(\log k \cdot \log(k/\varepsilon))$ . [34] and [19] can also achieve the same parameters by replacing the condenser in [13] with their condenser versions. On the other hand, [34] and [9] achieve subconstant entropy loss  $m = (1 - 1/\text{poly log } n)k$ ,  $d = O(\log n)$  when  $\varepsilon \geq 1/2^{\log^\beta n}$  for any constant  $\beta < 1$ .

In terms of computational complexity, an explicit construction is an algorithm that can compute the function in deterministic polynomial time on given parameters. A natural question is whether one can construct extractors in lower complexity classes, with matching parameters to the current best explicit ones. Some early work on extractors already pays attention to constructions in low-complexity models. For example, Zuckerman [41] showed that his construction is actually in  $NC$ . Also Bar-Yossef, Reingold, Shaltiel, and Vadhan [2] showed streaming constructions for several pseudorandom objects including extractors. Furthermore, extractors in low-complexity models have already been used in derandomization tasks for certain low-complexity classes, such as in [35, 8]. In this paper, we specifically focus on two low-complexity classes, i.e.  $AC^0$  and  $NC^1$ .  $AC^0$  is the class of all uniform circuit families of polynomial-size, constant depth, with NOT, AND, and OR gates, where AND and OR gates have unbounded fan-in.  $NC^1$  is the class of all uniform circuit families of polynomial-size,  $O(\log n)$  depth, with NOT, AND, and OR gates, where AND, OR gates have fan-in 2. Unless otherwise specified, our constructions are all logspace-uniform circuit families, i.e. there exists a logspace Turing machine that can output the description for each circuit in the family.

Viola [39] raised the question on extractor construction in  $AC^0$  and showed that for every constant  $D$ , there exists a polynomial  $p$  such that as long as  $k \leq n/p(\log n)$ , no extractor in  $AC^0$  with depth  $D$  extract even 1 bit with a constant error, no matter how long the seed is. Goldreich and Wigderson [10] extend the result for bit-fixing sources. This rules out the possibility for the case that  $k = n/\log^{\omega(1)} n$ . For the case  $k \geq n/\text{poly log } n$ , [10] gives a strong extractor in  $AC^0$  that has an output length linear to the seed length. Lately Cheng and Li [7] give a construction that significantly improves the parameters. For the case that  $\varepsilon = 1/\text{poly } n$ ,  $\delta = 1/\text{poly log } n$ , they achieve  $d = O(\log n)$ ,  $m = O(\delta n)$ . For the more general case that  $\varepsilon = 2^{-\text{poly log } n}$ ,  $\delta = 1/\text{poly log } n$ , they achieve  $d = O\left(\log n + \frac{\log(n/\varepsilon)\log(1/\varepsilon)}{\log n}\right)$ ,  $m = O(\delta n)$ . They also show that  $\varepsilon$  has to be at least  $2^{-\text{poly log } n}$  for  $AC^0$  extractors.

For extractors in  $NC^1$ , unlike the  $AC^0$  case, there are no known lower bounds for  $k$  or  $\varepsilon$ . Indeed the extractor based on universal hash functions [5], argued by the leftover hash lemma [16], can achieve an arbitrary  $\varepsilon$  and  $k$ . It can be realized in  $NC^1$  since there are simple linear function constructions for such hash functions. Trevisan's extractor [36], and its improved version [26] can also be realized in  $NC^1$ , since their main components, the average-case hard function based on local list-decodable codes can be computed in  $NC^1$ .

Extractors can also be derived from averaging samplers [41]. Healy [15] constructs a sampler in  $\text{NC}^1$ . However if one simply applies the transformation of [41] on it, then this can only give an extractor with a constant error. So it is still a question whether one can achieve extractors in  $\text{NC}^1$  with better parameters for arbitrary  $k$  and  $\varepsilon$ .

## 1.1 Our results

Our main positive result is an  $\text{AC}^0$  computable extractor with parameters optimal up to constant factors.

► **Theorem 1.** *For every constant  $a, c > 0, \gamma \in (0, 1)$ , every  $k \geq \frac{n}{\log^a(n)}, \varepsilon \geq 2^{-\log^c(n)}$ , there exists an explicit  $(k, \varepsilon)$ -strong extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  in  $\text{AC}^0$  with depth  $O((a + c + 1)^2)$ , such that  $d = O(\log \frac{n}{\varepsilon})$ , and  $m \geq (1 - \gamma)k$ .*

Notice that this is much better in seed length compared to the previous best  $\text{AC}^0$  constructions [7], which requires  $d = O\left(\left(\log n + \frac{\log(n/\varepsilon)\log(1/\varepsilon)}{\log n}\right) \log^a n\right)$  for such an output length. Also, notice that there are lower bounds for  $k$  and  $\varepsilon$  in the  $\text{AC}^0$  construction setting, i.e.  $k$  has to be at least  $n/\text{poly log } n$  by [10] and  $\varepsilon$  has to be  $2^{-\text{poly log } n}$  by [7]. Thus roughly in the plausible range for  $k$  and  $\varepsilon$ , we achieve parameters optimal up to constant factors.

Our method can also be used to give  $\text{NC}^1$  computable extractors.

► **Theorem 2.** *For every constant  $\gamma \in (0, 1)$  every  $k \geq \Omega(\log^2(n)), \varepsilon \geq 2^{-O(\sqrt{k})}$ , there exists a strong  $(k, \varepsilon)$  extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  computable in  $\text{NC}^1$ , with  $d = O(\log(n/\varepsilon))$ ,  $m = (1 - \gamma)k$ .*

To our knowledge, the previous best known  $\text{NC}^1$  construction is the improved Trevisan's extractor from [26], which has seed length  $O(\log^2 n \log \frac{1}{\varepsilon})$ , for all  $k, \varepsilon$ . Our parameters are optimal up to constant factors for ranges of  $k, \varepsilon$  as stated.

Our negative result generalizes the previous entropy parameter lower bound by [10] for strong extractors in  $\text{AC}^0$  to strong dispersers in  $\text{AC}^0$ .

► **Theorem 3.** *For every  $d, s > 0$ , every constant  $\delta \in (0, 1)$ , if  $C : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}$  is a  $(k, \frac{1}{2} - \delta)$ -disperser that can be computed by a non-uniform  $\text{AC}$  circuit of size  $s$  and depth  $d$ , then  $k \geq \Theta\left(\frac{\delta n}{\log^{d-1} s}\right)$ .*

## 1.2 Technique Overview

### 1.2.1 Extractor in $\text{AC}^0$

Our  $\text{AC}^0$  computable extractor is constructed by three main parts.

#### 1.2.1.1 Merger in $\text{AC}^0$

In this part, we show that any somewhere high-entropy source  $X$  can be merged to be a high-entropy source in  $\text{AC}^0$  under a restricted setting of parameters. The merger is a crucial building block in the construction of our extractor.

Recall that  $X = (X_1, \dots, X_\Lambda)$  is a simple somewhere  $(n, k)$  source if there exists  $i \in [\Lambda]$ ,  $X_i$  is a  $(n, k)$  source. We call each  $X_i$  a segment. A somewhere  $(n, k)$  source is a convex combination of simple somewhere  $(n, k)$  sources. A  $(k, k', \varepsilon)$  merger is a function  $\text{Merge} : \{0, 1\}^{n\Lambda} \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , such that for any input somewhere  $(n, k)$  source  $X$ ,  $\text{Merge}(X, U)$  has entropy  $k'$ . [9] gives a fairly good merger for somewhere uniform sources, which has  $m = n = k, k' = (1 - \delta)k, d = \frac{1}{\delta}(\log \frac{2\Lambda}{\varepsilon})$ . Our key observation is that if the

number of segments in the somewhere uniform source is  $\text{poly log } n$ ,  $\delta$  is a small constant, and error  $\varepsilon = 2^{-\text{poly log } n}$ , then this merger can be computed in  $AC^0$ . To see this, note that the computation of [9] is over a finite field  $F_q$ , where  $q = 2^d = 2^{\text{poly log } n}$  in this setting. The computation only involves three operations: (1) the summation of  $\text{poly log } n$  elements; (2) the powering  $y^i$  where  $y \in \mathbb{F}_q, i = \text{poly log } n$ ; (3) the product of a constant number of field elements. (1) is clearly in  $AC^0$  since it is actually the summation of  $\text{poly log } n$  bits, while (2) and (3) are shown to be in  $AC^0$  by [14]. Note that this can be straightforwardly generalized to a merger for somewhere high-entropy source by first applying an extractor to each segment and then merging them.

### 1.2.1.2 Error Reduction

In this part, we give a new error reduction that can be realized in a highly parallel way. The required seed length is optimal up to constant factors, significantly better than [7]. Our method takes the basic extractor from [7], applies error reduction and stretches the output length to  $\text{poly}(\log n)$  bits. The stretching is designed to satisfy the requirement in the next part.

Let  $X$  be an input  $(n, k)$ -source with  $k = n/\log^a n$  for some constant  $a$ . We start from an  $AC^0$  computable  $(k, \varepsilon_0)$  extractor  $\text{EXT}_0 : \{0, 1\}^n \times \{0, 1\}^{d_0} \rightarrow \{0, 1\}^{m_0}$  where  $\varepsilon_0 = 1/n$ ,  $d_0 = O(\log n)$ ,  $m_0 = O(k^2/n)$ , which is achieved in [7]. Then for every given constant  $c$ , the new error reduction can reduce the error to be as small as  $\varepsilon = 2^{-\log^c(n)}$ , with a seed length  $O(\log \frac{n}{\varepsilon})$ . We briefly describe the main steps of the procedure along with their arguments.

1. Apply  $\text{EXT}_0$  to  $X$  for  $t = \frac{\log(n/\varepsilon)}{\log n}$  times in parallel, using independent seeds, outputting  $Y_1, Y_2, \dots, Y_t$  respectively, each of length  $m_0$ .  
Notice that by the error reduction of [25], one can show that with probability at least  $1 - \varepsilon' \geq 1 - O(\varepsilon_0)^t$ , there exists  $i$  such that  $Y_i$  has min-entropy at least  $m_0 - O(\log t)$ , while the seed length used here is only  $td_0 = O(\log(n/\varepsilon))$ . Hence one can deduce that  $(Y_1, \dots, Y_t)$  is  $t\varepsilon'$  close to a somewhere  $(m_0, m_0 - O(\log t))$  source. We stress that this step is also the first step in the error reduction of [7]. But we differ from [7] after then.
2. For each  $i$ , cut  $Y_i$  into  $l = O(\log n)$  blocks such that their lengths form a geometric sequence. That is  $Y_i = (Y_{i,1}, Y_{i,2}, \dots, Y_{i,l})$ , where we let  $m_j = |Y_{i,j}| = m_0^{0.1} \cdot 3^j$ . Denote  $Y_{i,1\dots j}$  as the first  $j$  blocks of  $Y_i$ . Then for each  $j$ , let  $B_j = (Y_{1,1\dots j}, Y_{2,1\dots j}, \dots, Y_{t,1\dots j})$ , i.e. the  $i$ -th segment of  $B_j$  is the first  $j$  blocks from  $Y_i$ . Regard  $B_j$  as a somewhere high-entropy source and merge it by the merger from the previous part, attaining  $Z_j$ . Here we use the same seed for each  $j$ . Then we regard  $(Z_1, Z_2, \dots, Z_l)$  as a block source and extract in a standard way by using an extractor  $\text{EXT}_1$ . Here  $\text{EXT}_1$  is constructed by first sampling  $O(\log \frac{n}{\varepsilon})$  bits from the source and then applying universal hashing.  
Notice that since the high entropy segment of  $Y$  is a  $(m_0, m_0 - O(\log t))$  source, each  $B_j$  has to be a somewhere  $(M_j, M_j - O(\log t))$  source, where  $M_j = m_1 + m_2 + \dots + m_j$ . Also, as  $t = \text{poly log } n$ , the merger can be implemented in  $AC^0$ . As a result of merging,  $Z_j$  has a high constant entropy rate. Since  $m_j, j \in [l]$  forms a geometric sequence,  $Z_j$  is a constant times longer than  $Z_{j-1}$ . Thus  $(Z_1, Z_2, \dots, Z_l)$  is indeed very close to a block source that has a constant conditional entropy rate. The output length is  $\Omega(\log n \log \frac{n}{\varepsilon})$  since for each block we can sample  $O(\log \frac{n}{\varepsilon})$  bits and then apply an extractor from the left-over hash lemma. The seed length is  $O(\log \frac{n}{\varepsilon})$  since both the merger and the sample-then-extract have a seed length  $O(\log \frac{n}{\varepsilon})$ .
3. Assume the previous steps give an extractor  $\text{EXT}'$ . To increase the output length, we run the above steps again but instead use  $\text{EXT}'$  to replace  $\text{EXT}_1$  in the second step. This can increase the output length by a  $\Omega(\log n)$  factor. We do this for  $b$  times to finally get an extractor with output length  $\Omega(\log^b n \cdot \log \frac{n}{\varepsilon})$ , for a given arbitrary constant  $b$ .



Note that in this way the circuit depth has a factor  $b$  blow-up. The seed length also has a factor  $b$  blow-up. But as  $b$  is a constant, the construction is still in  $AC^0$  and the seed length is still  $O(\log \frac{n}{\varepsilon})$ .

### 1.2.1.3 Output Stretch

The last part is a new output stretch procedure for  $AC^0$  computable extractors. Compared to the one in [7], the new method attains an output length  $(1 - \gamma)k$  with a seed length  $O(\log \frac{n}{\varepsilon})$ .

Observe that if the input source already has a constant entropy rate, then this is an easy case. Because one can do sampling to get a two-block source with constant conditional entropy rates. Then one can use the extractor derived from the previous part to extract from the second source, attaining a  $\text{poly} \log \frac{n}{\varepsilon}$  length output, and then use it to extract the first block by applying the main extractor from [7]. However, the hard case is when the entropy rate is sub-constant i.e.  $k = \frac{n}{\log^a n}$ . The above simple strategy does not work since we don't know how to argue that the block attained from sampling can keep a constant fraction of all entropy while conditioned on this block, the source still keeps a fairly large conditional entropy. To resolve this issue, we follow a general strategy used in [9]. We describe the following 3 steps to reduce the hard case to the easy case.

1. Use Ta-shma's somewhere-block-source converter [33] to convert the original source into a somewhere-two-block-source.

Recall that Ta-shma's converter tries every position of the input source. For each position, the source is cut into two substrings. To avoid having too many segments in the resulting somewhere-two-block-source, one can pick a cutting position after, for example, every  $n / \log^{2a} n$  consecutive positions. In this way, the number of segments is  $\Lambda = \log^{2a} n$ . [33] shows that for at least one of the position choices, the cutting can give a two-block source where the first block has entropy  $\Omega(k)$ , and the second has conditional entropy  $\Omega(k)$ .

2. For each segment, apply our extractor in the error reduction part for the second block and then use the output as a seed to extract the first block by the extractor in [7].

As at least one segment of the somewhere source is indeed a two-block source, the extraction for the second block can provide an output of length  $\text{poly} \log \frac{n}{\varepsilon}$ . This is enough to extract a constant fraction of entropy i.e.  $\Omega(k)$  from the first block by [7]. Then what we get is very close to a somewhere uniform source.

3. Use the merger in  $AC^0$  from the previous part to get a source with a constant entropy rate and min-entropy  $\Omega(k)$ .

As we only have  $\text{poly} \log n$  segments,  $\varepsilon = 2^{-\text{poly} \log n}$ , and the entropy rate attained is a constant, it holds that the merger is in  $AC^0$ , with a seed length  $O(\log \frac{n}{\varepsilon})$ . Then after merging, the hard setting is reduced to the previously discussed easy setting, i.e. the constant entropy rate case.

### 1.2.2 Extractor in $NC^1$

Our construction for extractor in  $NC^1$  can be described by the following 3 steps.

1. First apply a condenser from [19]. Regard the output as  $(Y_1, Y_2)$  such that  $Y_1, Y_2$  have a equal length.

Compared to the condenser in [13], the condenser in [19] can only work for  $k \geq \Omega(\log^2(n)), \varepsilon \geq 2^{-O(\sqrt{k(n)})}$ . However, the advantage is that it is computable in  $NC^1$ . Recall that the [19]  $(k, k + d, \varepsilon)$  condenser can actually be viewed as  $\text{Cond} : \mathbb{F}_q^n \times \mathbb{F}_q \rightarrow \mathbb{F}_q^m$ . It views the input source as coefficients of a degree  $n - 1$  polynomial  $f(x) = \sum_{i=0}^{n-1} a_i x^i$

over field  $\mathbb{F}_q$ ,  $\log q = O(\log \frac{n}{\epsilon})$ . The seed is a random element of  $\mathbb{F}_q$ . The computation is actually  $\text{Cond}(f, u) = (u, f(u), f^{(1)}(u), \dots, f^{(m)}(u))$ . Where  $f^{(j)}(u) = \sum_{i=0}^d \frac{i!}{(i-j)!} a_i u^{i-j}$  is the  $j$ -th derivative of  $f$ . Notice that all these coefficients  $\frac{i!}{(i-j)!}$  can be precomputed and hardwired in the circuits. The polynomial evaluation consists of three operations: (1) the powering  $x^{i-j}$ , (2) the multiplication of two  $\mathbb{F}_q$  elements, and (3) the summation of a polynomial number of elements. The powering could be implemented with two steps: powering in  $\mathbb{N}$  and then divided by  $q$ , which is computable in  $NC^1$  by [4]. The multiplication and summation are both in  $NC^1$  by straightforward realizations. So after condensing, we get a source  $(Y_1, Y_2)$  with an entropy rate  $> 3/4$ . As  $Y_1$  and  $Y_2$  have an equal length, they form a two-block source with constant conditional entropy rates.

2. For  $Y_2$ , apply the extractor from our error reduction to get  $Z$  of length  $O(\log^2 n \log(n/\epsilon))$ . This step is basically the same as the  $AC^0$  case. We make sure the error reduction can also be done in  $NC^1$  under this parameter setting, and the seed length is still  $O(\log \frac{n}{\epsilon})$ .
3. Apply the improved Trevisan's extractor [26] to  $Y_1$  using  $Z$  as the seed.

Notice that this extracts  $O(k)$  bits with a desired error. It can be further stretched to  $(1 - \gamma)k$  by a standard parallel method. Also, notice that it is a folklore that Trevisan's extractor [36] and its improved version [26] can be realized in  $NC^1$ . So our whole construction is in  $NC^1$ . The required seed length for improved Trevisan's extractor is  $O(\log^2 n \log(n/\epsilon))$ , and the output from step 2 is enough to feed it. Hence the overall seed length is  $O(\log \frac{n}{\epsilon})$ .

### 1.2.3 A lower bound for $AC^0$ computable dispersers

Our lower bound follows from the improved switching lemma in [28]. Assume  $\text{Disp} : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}$  is a strong  $(k, \frac{1}{2} - \delta)$ -disperser computable in  $AC^0$  with depth  $d$  and size  $s$ . Notice that we only need to consider the 1 bit output setting. Consider that for a fixed seed  $y \in \{0, 1\}^r$ , we apply a random restriction on  $C_y := \text{Disp}(\cdot, y)$ . Let the random restriction be  $R_p$  over  $\{0, 1, *\}^n$  such that for every  $i \in [n]$ , independently we have  $\Pr[R_p(i) = *] = p, \Pr[R_p(i) = 0] = \Pr[R_p(i) = 1] = \frac{1-p}{2}$ . For a restriction  $\rho$  sampled from  $R_p$ , the function  $C_y|_\rho$  is defined to be a function such that if  $\rho_i$  is 1 or 0 then fix the  $i$ -th input to be  $\rho_i$ , otherwise leave it unfixed, and then apply  $C_y$  on this modified input. The switching lemma from [28] basically shows that  $\Pr_{\rho \sim R_p}[C_y|_\rho \text{ is not constant}] \leq \delta$ , if  $p = \frac{\delta}{\Theta((\log s)^{d-1})}$ . Also notice that when  $\delta$  is a constant, with probability at least  $1 - 2^{-O(pn)} > 1 - \delta$ , the number of stars in  $\rho$  is at least  $p/2$  fraction. By a union bound and an averaging argument, one can show that there exists a  $\rho$  which has at least  $pn/2$  stars such that for  $> 1 - 2\delta$  fraction of  $y$ ,  $C_y|_\rho$  is a constant. Notice that if we take this  $\rho$  for a uniform input source, then it becomes a bit-fixing source of entropy  $k \geq pn/2 = \Theta(\frac{\delta n}{(\log^{d-1} s)})$ . Also notice that for every  $y$  such that  $C_y|_\rho$  is not fixed,  $\text{Supp}(C_y|_\rho(X)) \leq 2$  as  $C_y$  only has 1 bit output. This implies that  $|\text{Supp}(U, \text{Disp}(X, U))|$  is less than  $2\delta 2^r \cdot 2 + (1 - 2\delta)2^r \leq (\frac{1}{2} + \delta)2^{r+1}$ , a contradiction to the disperser definition.

## 1.3 Paper Organization

In Section 2 we prepare some basic tools used in the rest of the paper. In Section 3 we show that merger can be implemented in  $AC^0$ . In Section 4 we give our new error reduction. In Section 5 we give our new output stretch and show our  $AC^0$  computable extractor finally. In Section 6 we show our  $NC^1$  computable extractor. In Section 7 we give our lower bound for dispersers in  $AC^0$ . In Section 8 we describe some open questions.

## 2 Preliminaries

We use the following results from previous works. First, we review the extractors in  $\text{AC}^0$  from [7]. They are actually logspace-uniform constructions, though [7] did not explicitly mention this. We briefly explain the reason after exhibiting their results.

► **Theorem 4** ([7]). *For every constant  $a, c \geq 1$ , every  $k = \delta n = \Theta(n/\log^a n)$  there exists an explicit  $(k, 1/n^c)$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  computable in  $\text{AC}^0$  with depth  $O(a)$ , where  $d = O(\log n)$ ,  $m = k^{0.01}$ .*

► **Remark 5.** Theorem 4 uses several tools and all of them can be implemented by logspace-uniform  $\text{AC}^0$  circuits. Specifically they use hardness amplifications from [17] and [18] and the Nisan-Wigderson (NW) generator [22]. These tools only use 4 kinds of operations: 1) pairwise independent generator; 2) inner product in  $\mathbb{F}_2^{O(\log n)}$ ; 3) parity function on  $O(\log n)$  bits; 4) Construct a combinatorial design and run the NW generator. It is straightforward to see that Procedure 1), 2) and 3) are all logspace-uniform. Procedure 4) is also logspace-uniform by Lemma A.3 in [6].

For smaller errors, they have the following theorem.

► **Theorem 6** ([7] for small entropy). *For every constant  $\gamma \in (0, 1)$ ,  $a, c \geq 1$ ,  $k = \delta n = \Theta(n/\log^a n)$ ,  $\varepsilon = 2^{-\Theta(\log^c n)}$ , there exists an explicit  $(k, \varepsilon)$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  in  $\text{AC}^0$  with depth  $O(a+c)$ , where  $d = O\left(\left(\log n + \frac{\log(n/\varepsilon)\log(1/\varepsilon)}{\log n}\right)/\delta\right)$ ,  $m \geq (1-\gamma)k$ .*

Also, recall the sample-then-extract technique in  $\text{AC}^0$ .

► **Theorem 7** ([7] Sample-then-extract). *For every constant  $\delta \in (0, 1]$ ,  $c \geq 1$  and every  $\varepsilon = 2^{-\log^c n}$ , there exists an explicit strong  $(\delta n, \varepsilon)$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  in  $\text{AC}^0$  with depth  $O(c)$ , where  $d = O(\log(n/\varepsilon))$ ,  $m = \Theta(\log(n/\varepsilon))$ .*

► **Remark 8.** Theorem 7 has two main ingredients: 1) The  $\text{NC}^1$  sampler from [15]. 2) Transforming a circuit of input length  $l = \Theta(\log^c n)$ , depth  $O(\log l)$  and size  $\text{poly}(l)$  to a  $\text{AC}^0$  circuit, from [12] (See also Lemma 12). Both of them are indeed logspace-uniform.

Theorem 6 uses Theorem 4 together with an error reduction and output stretch procedure. Both the error reduction and output stretch only consist of some sample-then-extract techniques and some utilities of the transformation from [12]. Hence it is also logspace-uniform.

Leftover hash lemma is also needed in our construction.

► **Lemma 9** (Leftover Hash Lemma [16]). *Let  $X$  be an  $(n', k = \delta n')$ -source. For any  $\Delta > 0$ , let  $H$  be a universal family of hash functions mapping  $n'$  bits to  $m = k - 2\Delta$  bits. The distribution  $U \circ \text{EXT}(X, U)$  is at distance at most  $1/2^\Delta$  to uniform distribution where the function  $\text{EXT} : \{0, 1\}^{n'} \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  chooses the  $U$ 'th hash function  $h_U$  in  $H$  and outputs  $h_U(X)$ .*

*For universal hash functions, we use the construction from Toeplitz matrices. For every  $u$ , the hash function  $h_A(x)$  equals to  $Ax$  where  $A$  is a Toeplitz matrix.*

Error reduction for extractors has been extensively studied in previous works. We recall the following key ingredient in the classic error-reducing technique [25].

► **Lemma 10** ( $G_x$  Property [25]). *Let  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  be a  $(k, \varepsilon)$ -extractor with  $\varepsilon < 1/4$ . Let  $X$  be any  $(n, k+t)$ -source. For every  $x \in \{0, 1\}^n$ , there exists a set  $G_x$  such that the following holds.*

## 69:8 Randomness Extractors in $AC^0$ and $NC^1$ : Optimal up to Constant Factors

- For every  $x \in \{0, 1\}^n$ ,  $G_x \subset \{0, 1\}^d$  and  $|G_x|/2^d = 1 - 2\epsilon$ .
- If we draw a  $y$  from  $\text{EXT}(X, G_X)$  (draw an  $x$  from  $X$ , then draw  $g_x$  uniformly from the set  $G_x$ , take  $y = \text{EXT}(x, g_x)$ ), then with probability at least  $1 - 2^{-t}$  over this random drawing, the  $y$  we get can have the property that  $\Pr[\text{EXT}(X, G_X) = y] \leq 2^{-(m-1)}$ . Here  $\text{EXT}(X, G_X)$  is obtained by first sampling  $x$  according to  $X$ , then choosing  $r$  uniformly from  $G_x$ , and outputting  $\text{EXT}(x, r)$ .

We also need to use the following lemmas about low-depth circuits computing.

► **Lemma 11** (folklore, see also [7]). *Let  $a > 0$  be an absolute constant. Then  $\log^a(n)$ -bit parity can be computed by an  $AC^0$  circuit with  $O(a)$  depth and  $\text{poly}(n)$  size.*

► **Lemma 12** ([12]). *For every  $c \in \mathbb{N}$ , every integer  $l = \Theta(\log^c n)$ , if the function  $f_l : \{0, 1\}^l \rightarrow \{0, 1\}$  can be computed by circuits of depth  $O(\log l)$  and size  $\text{poly}(l)$ , then it can be computed by  $AC^0$  circuits of depth  $c + 1$ , size  $\text{poly}(n)$ .*

► **Remark 13.** The transformation from [12] mainly uses Barrington's Theorem [3] which provides a Dlogtime-uniform  $AC^0$  reduction from any  $NC^1$  circuit to a downward self-reducible  $NC^1$ -complete language. The self-reducible here is logspace-uniform  $NC^0$  reduction. Thus the  $NC^1$  complete language of input size  $l = \Theta(\log^c n)$  can be reduced to a language of input size  $O(\log n)$  and thus can be decided by logspace-uniform  $AC^0$  circuits.

Finally, we use some folklore facts about block sources. Proofs of them can be found in the full version.

► **Definition 14** (block source). *Let  $X = (X_1, \dots, X_l)$  such that each  $X_i$  is distributed on  $\{0, 1\}^{n_i}$ . We say  $X$  is a  $(n_1, k_1, n_2, k_2, \dots, n_l, k_l)$ -block source if for every  $i \in [l]$  and  $(x_1, \dots, x_{i-1}) \in \{0, 1\}^{n_1 + \dots + n_{i-1}}$ ,  $X_i|_{X_1=x_1, \dots, X_{i-1}=x_{i-1}}$  is a  $(n_i, k_i)$ -source.*

► **Lemma 15.** *Fix  $t \in \mathbb{N}$  and  $k, s, n, n_1, \dots, n_k \in \mathbb{N}$  such that  $n_1 + \dots + n_k = n$ . Let  $X = (X_1, \dots, X_l)$  be a  $(n, n - k)$ -source on  $\{0, 1\}^n$  such that  $X_i$  is distributed on  $\{0, 1\}^{n_i}$  for each  $i \in [t]$ . Then  $(X_1, \dots, X_l)$  is  $l \cdot 2^{-s}$ -close to a  $(n_1, n_1 - k, n_2, n_2 - k - s, \dots, n_l, n_l - k - s)$ -source.*

► **Lemma 16.** *Let  $X = (X_1, \dots, X_l)$  be a  $(n_1, k_1, n_2, k_2, \dots, n_l, k_l)$ -block source on  $\{0, 1\}^n$ . Suppose that  $\text{EXT}_i : \{0, 1\}^{n_i} \times \{0, 1\}^r \rightarrow \{0, 1\}^{m_i}$  is a strong  $(k_i, \epsilon)$ -extractor for each  $i \in [l]$ . Let  $Y$  be a uniformly random variable on  $\{0, 1\}^r$ . Take  $Z = (Z_1, \dots, Z_l)$  such that  $Z_i = \text{EXT}_i(X_i, Y)$ . Then  $(Y, Z)$  is  $l \cdot \epsilon$ -close to uniform.*

► **Definition 17** (strong two-block extractor). *We say a function  $\text{EXT} : \{0, 1\}^{n_1} \times \{0, 1\}^{n_2} \times \{0, 1\}^r \rightarrow \{0, 1\}^m$  is a strong  $(k_1, k_2, \epsilon)$ -two-block extractor, if for any  $(k_1, k_2)$ -block-source  $X = (X_1, X_2)$  and independent uniform random distribution  $U_r$  on  $\{0, 1\}^r$ , the joint distribution  $(U_r, \text{EXT}(X_1, X_2, U_r))$  is  $\epsilon$ -close to uniform distribution on  $\{0, 1\}^r \times \{0, 1\}^m$ .*

► **Lemma 18.** *Let  $\text{EXT}_1 : \{0, 1\}^{n_1} \times \{0, 1\}^{m_1} \rightarrow \{0, 1\}^{m_2}$  be a  $(k_1, \epsilon_1)$ -strong extractor, and  $\text{EXT}_2 : \{0, 1\}^{n_2} \times \{0, 1\}^r \rightarrow \{0, 1\}^{m_1}$  be a  $(k_2, \epsilon_2)$ -strong extractor. Then the construction*

$$\text{EXT}(X_1, X_2, U_r) = \text{EXT}_1(X_1, \text{EXT}_2(X_2, U_r)) \quad (1)$$

*is a strong  $(k_1, k_2, \epsilon_1 + \epsilon_2)$ -two-block extractor*

### 3 Merger in $AC^0$

In this section, we will examine the merger construction in [9] and show that the merger can indeed be implemented in  $AC^0$  for some specific setting of parameters.

We start by defining somewhere- $(n, k)$  sources.

► **Definition 19** (somewhere- $(n, k)$  source). *Let  $X = (X_1, \dots, X_\Lambda)$  such that each  $X_i$  is distributed on  $\{0, 1\}^n$ . We say  $X$  is a simple somewhere- $(n, k)$  source with  $\Lambda$  segments if there exists  $i \in [\Lambda]$  such that  $X_i$  is a  $(n, k)$ -source on  $\{0, 1\}^n$ . We say  $X$  is a somewhere-uniform source if  $X$  is a convex combination of simple somewhere- $(n, k)$  sources.*

*If  $n = k$  in the above definition, which means that  $X_i$  is uniform, we say  $X$  is a somewhere-uniform source.*

A merger is a function that takes a somewhere-uniform source and a uniform random seed as input and outputs a  $(m, k')$ -source. The remaining entropy  $k'$  is usually less than the original entropy  $k$ .

► **Definition 20** (merger and strong merger). *We say  $\text{Merge} : \{0, 1\}^{\Lambda \cdot n} \times \{0, 1\}^r \rightarrow \{0, 1\}^m$  is a  $(k, k', \varepsilon)$ -merger if for any somewhere- $(n, k)$  source  $X = (X_1, \dots, X_\Lambda)$ , the distribution  $\text{Merge}(X, U_r)$  is  $\varepsilon$ -close to a  $k'$ -source. Here  $U_r$  is a independent uniform random distribution on  $\{0, 1\}^r$*

*Furthermore, if  $(U_r, \text{Merge}(X, U_r))$  is  $\varepsilon$ -close to  $(U_r, W)$ , we say  $\text{Merge}$  is a strong  $(k, k', \varepsilon)$ -merger. Here  $W$  is a distribution such that for all  $a \in \{0, 1\}^r$ ,  $W|_{U_r=a}$  is a  $k'$ -source.*

We examine the merger introduced in [9], and find that the merger can be implemented in  $AC^0$  if the number of segments is not too large.

► **Theorem 21** (merger in [9]). *For any constant  $a, c > 0$ ,  $\delta \in (0, 1)$ , let  $\Lambda(n) \leq \log^a(n)$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ . Then there exists explicit  $(n, \delta n, \varepsilon(n))$ -mergers  $\text{Merge} : \{0, 1\}^{\Lambda(n) \cdot n} \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^n$ . Here  $r(n) = O(\log(\frac{1}{\varepsilon}))$ .*

*Furthermore, the mergers can be implemented in  $AC^0$  with  $O(a + c + 1)$  depth and  $\text{poly}(n)$  size,*

The merger in [9] is defined as follows:

Define  $q = 2^s$  be a power of two which is decided later. Let  $\mathbb{F}_q$  be the finite field of order  $q$ . Let  $X = (X_1, \dots, X_\Lambda)$  be a somewhere-uniform-source with  $\Lambda$  segments. Regard each  $X_i$  as distributed on  $\mathbb{F}_q^K$  with  $K = \frac{n}{s}$ . Then

$$X_i = (X_{i,1}, \dots, X_{i,K}), \quad X_{i,j} \in \mathbb{F}_q. \quad (2)$$

Note that the uniform distribution on  $\mathbb{F}_q^K$  is equivalent to the uniform distribution on  $\{0, 1\}^n$ .

Take  $\gamma_1, \dots, \gamma_\Lambda$  be  $\Lambda$  unique points in  $\mathbb{F}_q$ . Let  $C_1, \dots, C_\Lambda$  be  $\Lambda$  unique polynomials in  $\mathbb{F}_q[x]$  of degree at most  $\Lambda - 1$ , such that  $C_i(\gamma_j) = 1$  if  $i = j$  and  $C_i(\gamma_j) = 0$  if  $i \neq j$ . Then the merger is defined as:

$$\text{Merge}(X, y) = \left( \sum_{i=1}^{\Lambda} C_i(y) X_{i,1}, \dots, \sum_{i=1}^{\Lambda} C_i(y) X_{i,K} \right), \quad (3)$$

where  $y \in \mathbb{F}_q$ .

## 69:10 Randomness Extractors in $AC^0$ and $NC^1$ : Optimal up to Constant Factors

► **Lemma 22** (merger in [9]). *For any constant  $\delta > 0$ , let  $q \geq (\frac{2\Lambda}{\varepsilon})^{1/\delta}$ . Then the function  $\text{Merge} : \mathbb{F}_q^{K \cdot \Lambda} \times \mathbb{F}_q \rightarrow \mathbb{F}_q^K$  is a  $(K \log q, k, \varepsilon)$ -merger, where  $k = (1 - \delta) \cdot K \cdot \log q$ .*

The condition  $q \geq (\frac{2\Lambda}{\varepsilon})^{1/\delta}$  is equivalent to  $r \geq \frac{1}{\delta} \log(\frac{2\Lambda}{\varepsilon})$ . When  $\Lambda = \log^a(n)$ ,  $\varepsilon = 2^{-\log^c(n)}$ , this requires  $r \geq \frac{2}{\delta} \log^c(n)$ . So we can pick  $r(n) = \min\{s \in \mathbb{N} \mid s \geq \frac{2}{\delta} \log^c(n), \exists d \in \mathbb{N}, s = 3 \cdot 2^d\}$ . As  $\delta$  is a constant,  $r(n) = O(\log^c(n)) = O(\log(\frac{1}{\varepsilon}))$ .

► **Lemma 23.** *For any constant  $a, c, \delta \in (0, 1)$ , let  $\Lambda(n) \leq \log^a n$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ . Define  $r(n) = \min\{s \in \mathbb{N} \mid s \geq \frac{2}{\delta} \log^c(n), \exists d \in \mathbb{N}, s = 3 \cdot 2^d\}$ ,  $q(n) = 2^{r(n)}$ ,  $K(n) = \frac{n}{r(n)}$ . Then the  $(n, \delta n, \varepsilon)$ -merger  $\text{Merge} : \{0, 1\}^{\Lambda(n) \cdot n} \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^n$  can be implemented in uniform  $AC^0$  with  $O(a + c + 1)$  depth and  $\text{poly}(n)$  size.*

To prove the lemma, we can express the  $\Lambda$  polynomials  $C_1, \dots, C_\Lambda$  by their  $\Lambda^2$  coefficients. That is:

$$C_i(y) = \sum_{j=1}^{\Lambda} c_{i,j} y^{j-1}, \quad c_{i,j} \in \mathbb{F}_q, \quad i \in [\Lambda].$$

These coefficients are not necessarily computable in  $AC^0$ . Instead, they can be pre-determined and stored in the circuit. Note that  $\Lambda = \log^a(n)$  and  $r_2(n) = O(\log^c(n))$ . Therefore it requires  $O(\log^c(n))$  bits to store one coefficient, and  $O(\log^{2a+c}(n))$  bits to store all the coefficients.

Therefore, the  $AC^0$  circuit for the merger is only required to do three types of operations: powering, multiplication and summation. The parameters of these operations satisfies the following conditions:

1. The powering operation is to compute  $y^j$ , where  $j \leq \log^a(n)$ , and  $y \in \mathbb{F}_q$ . The order  $q = 2^s$  is a power of 2, and  $s = O(\log^c(n))$ .
2. The multiplication operation is to compute  $c_{i,j} y^{j-1} X_{i,k}$ , for each  $i \in [\Lambda], j \in [\Lambda], k \in [K]$ . All of the three multipliers are in  $\mathbb{F}_q$ .
3. The summation operation is to compute  $\sum_{i=1}^{\Lambda} \sum_{j=1}^{\Lambda} c_{i,j} y^{j-1} X_{i,k}$  for each  $k \in [K]$ . All the addends are in  $\mathbb{F}_q$ , and the total number of them is  $\log^{4a}(n)$ .

The following theorems in the work of Healy and Viola [14] show that the powering and multiplication are indeed in  $AC^0$ .

► **Lemma 24** ([14, Corollary 6(1)]). *Let  $a, c > 0$  be absolute constants. Let  $y \in \mathbb{F}_q$  where  $q = 2^s$  and  $s = 2 \cdot 3^d$  for some  $d \in \mathbb{N}$ . Suppose that  $j \leq \log^a(n)$  and  $s \leq \log^c(n)$ , then  $y^j$  can be computed by a logspace-uniform  $AC^0$  circuit with  $O(a + c)$  depth and  $\text{poly}(n)$  size.*

► **Lemma 25** ([14, Corollary 6(2)]). *Let  $a, c > 0$  be absolute constants. Let  $y_1, y_2 \in \mathbb{F}_q$  where  $q = 2^s$  and  $s = 2 \cdot 3^d$  for some  $d \in \mathbb{N}$ . Suppose that  $s \leq \log^c(n)$ , then  $y_1 \cdot y_2$  can be computed by a logspace-uniform  $AC^0$  circuit with  $O(c)$  depth and  $\text{poly}(n)$  size.*

The summation operation is also in  $AC^0$ , as the summation of elements in  $\mathbb{F}_q$  where  $q = 2^s$  is equivalent to bitwise parity of the binary representation of the elements if we implement  $\mathbb{F}_q$  by polynomial fields with coefficients in  $\mathbb{F}_2$ . When the number of addends is  $\text{poly} \log n$ , it is in  $AC^0$  by Lemma 11.

With these results, the merger can be implemented in  $AC^0$  with  $O(a + c)$  depth and  $\text{poly}(n)$  size.

**Proof of Lemma 23.** It is sufficient prove that each  $\sum_{i=1}^{\Lambda} \sum_{j=1}^{\Lambda} c_{i,j} y^{j-1} X_{i,k}$  can be computed in  $\text{AC}^0$  with  $O(a+c)$  depth and  $\text{poly}(n)$  size. The powering could be computed in  $O(a+c)$  depth and  $\text{poly}(n)$  size by Lemma 24. The multiplication could be computed in  $O(c)$  depth and  $\text{poly}(n)$  size by Lemma 25. The summation could be computed in  $O(a)$  depth and  $\text{poly}(n)$  size by Lemma 11.  $\blacktriangleleft$

Theorem 21 follows directly from Lemma 22 and Lemma 23.

**Proof of Theorem 21.** Take  $r(n) = \min\{s \in \mathbb{N} | s \geq \frac{2^{\log^c(n)}}{\delta}, \exists d \in \mathbb{N}, s = 3 \cdot 2^d\}$ ,  $q(n) = 2^{r(n)}$ ,  $K(n) = \frac{n}{r(n)}$  as discussed above. By Lemma 22, we know that the merger is a  $(n, k(n), \varepsilon(n))$ -merger, where  $k(n) = (1-\delta)n$ . By Lemma 23, we know that the merger can be implemented in  $\text{AC}^0$  with  $O(a+c)$  depth and  $\text{poly}(n)$  size.  $\blacktriangleleft$

As noted in [9], their merger for somewhere uniform sources can be extended to handle somewhere high entropy sources. Following their idea, we also prepare a merger for somewhere high entropy sources, and furthermore, it is computable by low-depth circuits.

► **Corollary 26.** *Let  $\delta \in (0, 1)$ ,  $\Lambda(n) \leq \text{poly}(n)$ ,  $\varepsilon(n) = 2^{-O(n)}$ ,  $\Delta(n) = O(\log(\frac{n}{\varepsilon}))$ . Then there exists a strong  $(n - \Delta(n), \delta m(n), \varepsilon(n))$ -merger  $\text{Merge} : \{0, 1\}^{\Lambda(n) \cdot n} \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$ . Here  $r(n) = O(\log(\frac{n}{\varepsilon}))$  and  $m(n) = \Omega(n)$ . The merger is computable in logspace-uniform  $\text{AC}^0[2]$ .*

*If  $\Lambda(n) \leq \log^a(n)$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$  for constant  $a, c > 0$ , then the merger can be implemented in  $\text{AC}^0$  with  $O(a+c+1)$  depth and  $\text{poly}(n)$  size.*

## 4 Error Reduction

The main theorem of this section is the following:

► **Theorem 27.** *For any constant  $a, c > 0$ ,  $b \in \mathbb{N}^+$ , every  $k(n) \geq n/\log^a(n)$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ , there exists a strong  $(k(n), \varepsilon(n))$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$ , where  $r(n) = O(\log(\frac{n}{\varepsilon(n)}))$ ,  $m(n) = \Theta(\log^b(n) \cdot \log(\frac{n}{\varepsilon(n)}))$ .*

*Furthermore, the extractor can be implemented in  $\text{AC}^0$  with  $O(b(a+c+1))$  depth.*

We show this theorem by giving a new error reduction stated as the following. To describe it, We fix  $a > 0$  to be a constant and  $k(n) = \frac{n}{\log^a n}$ .

► **Lemma 28.** *For any  $\varepsilon_0 \in (0, 1)$  every constant  $c > 0$  and  $\varepsilon = 2^{-\log^c n}$ , suppose there exists a  $(k, \varepsilon_0)$ -extractor  $\text{EXT}_0 : \{0, 1\}^n \times \{0, 1\}^{d_0} \rightarrow \{0, 1\}^{m_0}$  with  $m_0 \geq k^{0.01}$  and a family of strong  $(n_1/100, \varepsilon)$ -extractors  $\text{EXT}_1 : \{0, 1\}^{n_1} \times \{0, 1\}^{d_1} \rightarrow \{0, 1\}^{m_1}$  for every  $n_1 \in [m_0^{0.1}, m_0]$ , Then for any  $\varepsilon = 2^{-\log^c n}$ , there exists a strong  $(k, \varepsilon)$ -extractor  $\text{EXT}' : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , where  $d = O(d_1 + d_0 \cdot \frac{\log \varepsilon}{\log \varepsilon_0})$ ,  $m = \Theta(m_1 \cdot \log n)$ .*

*If  $\text{EXT}_0$  and  $\text{EXT}_1$  can be realized by depth  $h$  and  $g$  AC circuits respectively, then  $\text{EXT}'$  can be realized by a depth  $O(h+g+c+1)$  AC circuit.*

Now we describe the construction and analysis of Lemma 28.

### 4.1 Step 1: extracting in parallel

We apply  $\text{EXT}_0$  for  $t = \frac{\log(1/\varepsilon)}{\log(1/\varepsilon_0)}$  times in parallel, with independent seeds. Specifically, take  $U_{1,i}$  be independent uniform seeds in  $\{0, 1\}^{d_0}$  for every  $i \in [t]$ . Let  $Y = (Y_1, Y_2, \dots, Y_t)$ , where  $Y_i = \text{EXT}_0(X, U_{1,i})$ .

The step can be computed by depth  $h$  AC circuits because the extractor  $\text{EXT}_0$  has depth  $h$ , and the parallel extraction can be done without increasing the depth.

### Analysis

We now show that  $Y$  is close to a somewhere- $(m_0(n), m_0(n) - O(\log t))$ -source. The main idea is that by Lemma 10, we know that with high probability, at least one of the seeds  $U_i$  lands in  $G_x$ , which makes  $Y_i$  a good source with a high entropy rate. The following lemma states this formally:

► **Lemma 29.** *Let  $\text{EXT}_0 : \{0, 1\}^n \times \{0, 1\}^{d_0} \rightarrow \{0, 1\}^{m_0}$  be an  $(k, \varepsilon_0)$ -extractor and  $X$  be a  $(n, k + s)$ -source. Take independent seeds  $U_1, U_2, \dots, U_s \in \{0, 1\}^{d_0}$ . Let  $Y = (Y_1, Y_2, \dots, Y_t)$ , where  $Y_i = \text{EXT}_0(X, U_i)$ . Then  $Y$  is  $(2\varepsilon_0)^t + t \cdot 2^{-s}$ -close to a somewhere- $(m_0, m_0 - O(\log t))$ -source*

Take  $x$  from a fixed distribution  $X$  and fix extractor  $\text{EXT}$ . Let  $G_x$  be the set of good seeds from Lemma 10. We first denote event  $\text{BAD}_i = \{U_i \notin G_x\}$ . Note that these events are not necessarily independent. However, the probability that all of them happen is exponentially small, as the following claim shows.

▷ **Claim 30.**  $\Pr[\text{BAD}_1 \wedge \text{BAD}_2 \wedge \dots \wedge \text{BAD}_t] \leq (2\varepsilon_0)^t$ .

We define an indicator random variable  $I \in \{0, 1\}^{[t]}$  as follows:

$$\forall i \in [t], i \in I \iff U_i \in G_x. \quad (4)$$

With probability at least  $1 - (2\varepsilon_0)^t$ , The set  $I$  is not an empty set. Take  $Y_i = \text{EXT}(X, U_i)$ . By Lemma 10,  $Y_i|_{(\text{BAD}_i)^c} = Y_i|_{i \in I}$  is  $2^{-s}$ -close to a  $(m_0, m_0 - O(1))$  source.

We apply the technique from [20] to prove that  $(Y_1, Y_2, \dots, Y_t)$  is indeed close to a somewhere- $(m_0, m_0 - O(\log t))$ -source.

► **Lemma 31** ([20]). *Let  $Y = (Y_1, \dots, Y_t)$  be the random variable defined in Lemma 29. Let  $I$  be a random set subset of  $[t]$ . Assume  $I \neq \emptyset$ , and for every  $i \in [t]$ ,  $Y_i|_{i \in I}$  is  $\varepsilon$ -close to a  $(m, k)$ -source. Then  $Y$  is  $(t \cdot \varepsilon)$ -close to a somewhere- $(m, k - \log t)$  source.*

By Claim 30 and Lemma 31, we can prove Lemma 29:

**Proof of Lemma 29.** Take  $I$  as the random set indicator defined above. By Lemma 10,  $Y_i|_{(\text{BAD}_i)^c} = Y_i|_{i \in I}$  is  $2^{-s}$ -close to a  $(m_0, m_0 - O(1))$  source. By Claim 30, we know that with probability at least  $1 - (2\varepsilon_0)^t$ ,  $I$  is not an empty set. Conditioning on such events, Lemma 31 implies that  $Y|_{\{I \neq \emptyset\}}$  is  $t \cdot 2^{-s}$ -close to a somewhere- $(m_0, m_0 - O(\log t))$  source. The lemma follows. ◀

## 4.2 Step 2: divide and merge

Assume we have a somewhere- $(m_0, m_0 - O(\log t))$ -source. We divide each segment of the source into a sequence of blocks whose lengths form a geometric sequence. Specifically, take  $Y = (Y_1, Y_2, \dots, Y_t)$  to be a simple somewhere- $(m_0, m_0 - O(\log t))$ -source. We divide each  $Y_i$  into  $l + 1$  blocks of length  $m_1, m_2, \dots, m_{l+1}$  respectively, such that

$$Y_i = (Y_{i,1}, Y_{i,2}, \dots, Y_{i,l+1}) \text{ for every } i \in [t]. \quad (5)$$

The lengths satisfies

$$m_j = m_0^{0.1} \cdot 3^{j-1} \text{ for every } j \in [l]. \quad (6)$$



where  $l = \lceil \log_3 m_0^{0.9} \rceil$ . Denote  $Y_{i,1\dots j} = (Y_{i,1}, Y_{i,2}, \dots, Y_{i,j})$  for every  $i \in [t]$  and  $j \in [l]$ . Define  $B_j$  as:

$$B_j = (Y_{1,1\dots j}, Y_{2,1\dots j}, \dots, Y_{t,1\dots j}) \text{ for every } j \in [l]. \quad (7)$$

We denote  $M_j = m_1 + m_2 + \dots + m_j$  for every  $j \in [l]$ .

Let  $\text{Merge}_j : \{0, 1\}^{t \cdot M_j} \times \{0, 1\}^{d_2(n)} \rightarrow \{0, 1\}^{(1-\alpha)M_j}$  be a strong  $(M_j - \Delta, \frac{3}{4}(1-\alpha)M_j, \varepsilon(n)/l)$ -merger from Corollary 26 for every  $j \in [l]$ , where  $\alpha$  is a constant. The seed length of the merger is  $d_2(n) = O(\log(\frac{M_j}{\varepsilon(n)})) = O(\log(\frac{m(n)}{\varepsilon(n)}))$ . Let  $U_2$  be a uniform random variable on  $\{0, 1\}^{d_2(n)}$ . Define

$$Z_j = \text{Merge}_j(B_j, U_2) \text{ for every } j \in [l]. \quad (8)$$

The gap between source length and source entropy is  $\Delta = O(\log t) = O(\log \frac{1}{\varepsilon(n)})$ , which meets the requirement that  $\Delta = O(\log \frac{M_j}{\varepsilon(n)})$  in Corollary 26.

Next, we apply the strong extractor family  $\text{EXT}_1$  to extract from the block source. Let  $\text{EXT}_{1,j} : \{0, 1\}^{(1-\alpha)M_j} \times \{0, 1\}^{d_3(n)} \rightarrow \{0, 1\}^{m'(n)}$  be a strong  $((1-\alpha)M_j/100, \varepsilon(n)/l)$ -extractor for every  $j \in [l]$ . These  $\text{EXT}_{1,j}, j \in [l]$  with different input lengths, are all from the family  $\text{EXT}_1$ . Let  $U_3$  be a uniform random variable on  $\{0, 1\}^{d_3(n)}$ . Then

$$W_j = \text{EXT}_{1,j}(Z_j, U_3) \text{ for every } j \in [l]. \quad (9)$$

### Analysis

Now we give our analysis. Note that since  $Y$  is a simple somewhere high entropy source, by dividing it into blocks, each prefix  $B_j$  is a simple somewhere- $(M_j, M_j - O(\log t))$ -source. Through merging,  $Z_j$ 's are correlated high-entropy sources with different lengths. They are close to a block source.

► **Lemma 32.**  $Z_j$  is  $\varepsilon(n)/l$ -close to a  $((1-\alpha)M_j, \frac{3}{4}(1-\alpha)M_j)$ -source for every  $j \in [l]$ .

**Proof.** Let  $Y_i$  be a  $(m_0, m_0 - O(\log t))$ -source in  $Y$ . Then  $Y_{i,1\dots j}$  must have entropy at least  $m_j - O(\log t)$ . Therefore  $B_j$  is a somewhere- $(m_j, m_j - O(\log t))$ -source. By Corollary 26,  $Z_j$  is  $\varepsilon(n)/l$ -close to a  $((1-\alpha)M_j, \frac{3}{4}(1-\alpha)M_j)$ -source. The claim follows. ◀

Denote  $Z_0 = (U_1, U_2)$  as the seeds used in all previous steps to obtain  $Z_1, \dots, Z_j$ . We stress that the sequence  $Z_0, Z_1, \dots, Z_l$  is of exponentially increasing length and each contains  $|Z_j| - O(\log \frac{1}{\varepsilon(n)})$  bits of min-entropy. Therefore, even if all the randomness in  $(Z_0, \dots, Z_i)$  is contained in  $Z_{i+1}$ , there still must be  $\Omega(|Z_{i+1}|)$  bits of conditional min-entropy within  $Z_{i+1}$ . That makes the sequence a block source. We formalize the inspection into the following lemma.

► **Lemma 33.**  $(Z_0, Z_1, Z_2, \dots, Z_l)$  is  $2\varepsilon(n)$ -close to a block source  $(Z_0, Z'_1, Z'_2, \dots, Z'_l)$ . The conditional entropy of  $Z'_j$  is larger than  $(1-\alpha)M_j/100 = \Omega((1-\alpha)M_j)$  for each  $j \in [l]$

Then we can extract from the block-source  $(Z_0, Z_1, Z_2, \dots, Z_l)$  using standard methods, which gives  $(Z_0, U_3, W_1, W_2, \dots, W_l)$ :

► **Lemma 34.**  $(Z_0, U_3, W_1, W_2, \dots, W_l)$  is  $3\varepsilon(n)$ -close to  $(Z_0, U_3, V)$ , where  $V$  is a independent uniform distribution.

### 4.3 Wrap-up to prove Lemma 28 and Theorem 27

**Proof of Lemma 28.** Take  $X$  be the sources,  $U_1, U_2, U_3$  be the seeds. Let  $Y = (Y_1, Y_2, \dots, Y_t)$  such that  $Y_i = \text{EXT}_0(X, U_{1,i})$  for every  $i \in [t]$  as in the first step. By Lemma 29,  $Y$  is  $\varepsilon(n)$ -close to a somewhere- $(m(n), m(n) - O(\log t))$ -source. Let  $B_j$  be the source  $(Y_{1,1\dots j}, Y_{2,1\dots j}, \dots, Y_{t,1\dots j})$  for every  $j \in [l]$ . Then take  $Z_j = \text{Merge}_j(B_j, U_2)$  and  $W_j = \text{EXT}_{1,j}(Z_j, U_3)$  for every  $j \in [l]$  as in the second step. Here  $\text{EXT}_{1,j}$  is the strong extractor from family  $\text{EXT}_1$  with source length  $n_1 = m_j$ . By Lemma 34 and its remark,  $(U_1, U_2, U_3, W)$  is  $3\varepsilon(n)$ -close to uniform if  $Y$  is a somewhere- $(m(n), m(n) - O(\log t))$ -source. By the triangle inequality,  $W$  is  $4\varepsilon(n)$ -close to uniform.

Step 1 executes the extractor  $\text{EXT}_0$  in parallel, which costs depth  $h$ . Step 2 executes the merger  $\text{Merge}_j$  from Corollary 26 and the extractor  $\text{EXT}_{1,j}$ , for every  $j \in [l]$  in parallel. This takes depth  $O(c + g)$ . So the overall depth is as the lemma stated.

The seed length of the extractor is  $d(n) = |U_1| + |U_2| + |U_3|$ .  $U_1 = (U_{1,1}, U_{1,2}, \dots, U_{1,t})$  where  $|U_{1,i}| = d_0$  for every  $i \in [t]$  and  $t = \frac{\log \varepsilon(n)}{\log \varepsilon_0}$ .  $|U_2| = O(\log(\frac{n}{\varepsilon(n)}))$  and  $|U_3| = d_1$ . Therefore  $d = O(d_1 + d_0 \cdot \frac{\log \varepsilon}{\log \varepsilon_0})$ .

The output consists of  $\Theta(\log n)$  parts of length  $m_1$ . Therefore the output length is  $m = \Theta(m_1 \cdot \log n)$ .  $\blacktriangleleft$

We instantiate  $\text{EXT}_0$  as the extractor from Theorem 4 and  $\text{EXT}_1$  as the strong extractors from Theorem 7, which gives the following theorem:

► **Corollary 35.** *For any constant  $a, c > 0$ , every  $k(n) \geq n/\log^a(n)$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ , there exists a strong  $(k(n), \varepsilon(n))$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$ , where  $r(n) = O(\log(\frac{n}{\varepsilon(n)}))$ ,  $m(n) = \Theta(\log(n) \cdot \log(\frac{n}{\varepsilon(n)}))$ .*

*Furthermore, the extractor can be implemented in uniform  $\text{AC}^0$  with  $O(a + c + 1)$  depth.*

The only gap between Corollary 35 and Theorem 27 is that the output length of Corollary 35 is only  $\Theta(\log(n) \cdot \log(\frac{n}{\varepsilon}))$  instead of  $\Theta(\log^b(n) \cdot \log(\frac{n}{\varepsilon}))$ . We resolve the issue by repeatedly using Lemma 28, each time instantiating  $\text{EXT}_1$  in Lemma 28 as the strong extractor family provided by the immediate previous using of Lemma 28. After an iteration, the output length is multiplied by a  $\Theta(\log n)$  factor. Therefore we can achieve the parameter as in Theorem 27 after  $b$  iterations.

## 5 Output Stretch

In this section, we will use the framework introduced in [9], to further stretch the output length from  $O(\log^c(n))$  to a near-optimal  $O(k)$ . The main theorem of this section is the following:

► **Theorem 36.** *For any constant  $a, c > 0$  and  $\gamma \in (0, 1)$ , let  $k(n) \geq \frac{n}{\log^a(n)}$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ . Then there exists a  $(k(n), \varepsilon(n))$ -strong extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$ , such that  $r(n) = O(\log(\frac{n}{\varepsilon}))$ , and  $m(n) \geq (1 - \gamma) \cdot k(n)$ .*

*Furthermore, the extractor can be implemented in  $\text{AC}^0$ , with  $O(a + c + 1)^2$  depth and  $\text{poly}(n)$  size.*

We use a four-step method to extract randomness.

## 5.1 Step 1: Converting to a somewhere-block-source

In this subsection, we will convert the original  $k$ -source into a somewhere-block-source. First, we define the concept:

► **Definition 37** (somewhere-block-source). *Let  $X = (X_1, \dots, X_\Lambda)$  be a random variable with  $\Lambda$  segments, each  $X_i$  distributed on  $\{0, 1\}^{n_1} \times \{0, 1\}^{n_2}$ . We say  $X$  is a simple  $(k_1, k_2)$ -somewhere-block-source if there exists  $i \in [\Lambda]$  such that  $X_i$  is a  $(k_1, k_2)$ -block-source. We say  $X$  is a  $(k_1, k_2)$ -somewhere-block-source if  $X$  is a convex combination of simple  $(k_1, k_2)$ -somewhere-block-sources.*

Ta-shma's somewhere-block-source converter [33] is a deterministic function that converts a  $k_1 + k_2 + s$ -source into a  $(k_1 - O(n/\Lambda), k_2)$ -somewhere-block-source, which has  $\Lambda$  segments.

Take  $X_1 \in \{0, 1\}^n$  as the original source, assume  $n$  is divisible by  $\Lambda$ , otherwise pad  $X_1$  with 0's. Regard  $X_1$  as a source with  $\Lambda$  parts, each of length  $n/\Lambda$ :

$$X_1 = (X_{1,1}, \dots, X_{1,\Lambda}) \in \left(\{0, 1\}^{n/\Lambda}\right)^\Lambda. \quad (10)$$

Now define the following separation of these parts into  $(Y_i, Z_i)$ :

$$Y_i = (X_{1,1}, \dots, X_{1,i}, 0^{(\Lambda-i) \cdot (n/\Lambda)}), \quad (11)$$

$$Z_i = (0^{i \cdot (n/\Lambda)}, X_{1,i+1}, \dots, X_{1,\Lambda}). \quad (12)$$

Then  $(Y_i, Z_i) \in \{0, 1\}^{2n}$ . The Ta-shma's somewhere-block-source converter is defined as the collection of all  $(Y_i, Z_i)$ , for  $i \in [\Lambda]$ :

$$B_{TS}^\Lambda(X_1) = \{(Y_i, Z_i) \in \{0, 1\}^{2n} \mid i \in [\Lambda]\}. \quad (13)$$

► **Theorem 38** ([33]). *Let  $\Lambda$  be an integer and  $\Lambda$  divides  $n$ . Let  $B_{TS}^\Lambda$  be the Ta-shma's somewhere-block-source converter defined above. Fix  $k, k_1, k_2, s \in \mathbb{N}$  such that  $k = k_1 + k_2 + s$ . Then for any  $k$ -source  $X \in \{0, 1\}^n$ ,  $B_{TS}^\Lambda(X)$  is  $O(n \cdot 2^{-s/3})$ -close to a  $(k_1 - O(n/\Lambda), k_2)$ -somewhere-block-source.*

Now we summarize the first step:

Step 1: Set  $\Lambda = \log^{2a}(n)$ , Take  $X_2 = (X_{2,1}, \dots, X_{2,\Lambda}) = B_{TS}^\Lambda(X_1)$  as a somewhere-block-source.

► **Lemma 39.** *For any constant  $a \geq 0$ , let  $k \geq \frac{n}{\log^a(n)}$ . Then for any  $k$ -source  $X_1 \in \{0, 1\}^n$ , the somewhere-block-source  $X_2 = B_{TS}^\Lambda(X_1)$  is  $n \cdot 2^{-\frac{n}{\log^{2a} n}}$ -close to a  $(k - O(\frac{n}{\log^{2a} n}), \frac{n}{\log^{2a} n})$ -somewhere-block-source.*

The first step can be computed in  $AC^0$  with  $O(1)$  depth and  $\text{poly}(n)$  size, as it is only splitting the input into blocks.

## 5.2 Step 2: Extracting from a somewhere-block-source

In this subsection, we focus on the good block of the somewhere-block-source, and extract randomness from it. A two-block extractor is employed in this section. We use the block-extraction technique together with our extractors from Theorem 6 and Theorem 27 to extract  $O(\log^{a+c} n)$  randomness from the second block of the block source, then use it as seed for another extractor, in order to extract  $O(k)$  randomness from the first block of the block source.

## 69:16 Randomness Extractors in $AC^0$ and $NC^1$ : Optimal up to Constant Factors

For a somewhere-block-source, we may apply the two-block extractor to each segment such that the good segment is converted into a somewhere-close-to-uniform source. The source is defined as follows:

► **Lemma 40.** *Let  $X = (X_1, \dots, X_\Lambda)$  be a  $(k_1, k_2)$ -somewhere-block-source, where each segment is a source on  $\{0, 1\}^{n_1} \times \{0, 1\}^{n_2}$ . Let  $EXT: \{0, 1\}^{n_1} \times \{0, 1\}^{n_2} \times \{0, 1\}^r \rightarrow \{0, 1\}^m$  be a  $(k_1, k_2, \varepsilon)$ -strong-two-block extractor. Let  $U_r$  be a uniform random distribution on  $\{0, 1\}^r$ . Then*

$$(EXT(X_1, U_r), \dots, EXT(X_\Lambda, U_r))$$

is  $\varepsilon$ -close to a somewhere-uniform-source.

Take  $EXT_1$  from Theorem 6 and  $EXT_2$  from Theorem 27. Construct  $EXT(X_1, X_2, U_r) = EXT_1(X_1, EXT_2(X_2, U_r))$  as a strong-two-block extractor. We have the following theorem:

► **Theorem 41 (block-extraction in  $AC^0$ ).** *There exists a constant  $\gamma \in (0, 1)$ . For any constant  $a, c > 0$ , let  $k_1(n) \geq \frac{n}{\log^a(n)}$ ,  $k_2(n) \geq \frac{n}{\log^{2a}(n)}$ ,  $\varepsilon(n) \geq 2^{-\log^c(n)}$ , there exists a  $(k_1(n), k_2(n), \varepsilon(n))$ -strong-two-block extractor  $EXT: \{0, 1\}^n \times \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}^m$ , such that  $r(n) = O(\log(\frac{n}{\varepsilon}))$ , and  $m(n) \geq (1 - \gamma)k_1(n)$ .*

Furthermore, the extractor can be implemented in  $AC^0$ , with  $O(a + c + 1)^2$  depth and  $\text{poly}(n)$  size.

We summarize the second step here:

Step 2: Take  $EXT: \{0, 1\}^n \times \{0, 1\}^n \times \{0, 1\}^{r_1(n)} \rightarrow \{0, 1\}^{m(n)}$  as a  $(\frac{n}{\log^a(n)}, \frac{n}{\log^{2a}(n)}, \varepsilon(n))$ -strong-two-block extractor, where  $r_1(n) = O(\log(\frac{n}{\varepsilon}))$  and  $m(n) \geq (1 - \gamma)k(n)$ . Take  $X_3 = (EXT(X_{2,1}, U_{r_1}), \dots, EXT(X_{2,\Lambda}, U_{r_1}))$  be  $2 \cdot \varepsilon(n)$ -close to a somewhere-uniform-source.

This step can be implemented in  $AC^0$  with  $O(a + c)$  depth and  $\text{poly}(n)$  size, as it is applying  $AC^0$  functions to each block of the input.

The source  $X_3$  is now  $\varepsilon(n)$ -close to a somewhere-uniform-source. It has  $\Lambda = \log^{2a}(n)$  segments, each of length  $m(n) \geq (1 - \gamma)k(n)$ . The next step is using the merger introduced in [9] to merge the segments into one source.

### 5.3 Step 3: Merging the segments

We use the merger introduced in [9] to merge the segments of the somewhere-uniform-source into one source. The construction of the merger is discussed in Theorem 21.

Step 3: Take  $Merge: \{0, 1\}^{\Lambda \cdot m(n)} \times \{0, 1\}^{r_2(n)} \rightarrow \{0, 1\}^{m(n)}$  be the  $(m(n), \frac{3}{4}m(n), \varepsilon(n))$ -merger from Theorem 21. Then  $X_4 = Merge(X_3, U_{r_2})$ .

As a direct consequence of Theorem 21 we have the following lemma.

► **Lemma 42.**  *$X_4$  is  $3 \cdot \varepsilon(n)$ -close to a  $\frac{3}{4}m(n)$ -source.*

Also, notice that the computation in  $AC^0$  with depth  $O(a + c)$ , with seed length  $O(\log(n/\varepsilon(n)))$ .

## 5.4 Step 4: Second extraction

The final step is as the following.

Step 4: Take  $\text{EXT}_2 : \{0, 1\}^{m(n)/2} \times \{0, 1\}^{m(n)/2} \times \{0, 1\}^{r_3(n)} \rightarrow \{0, 1\}^{m'(n)}$  be the  $(\frac{1}{8}m(n), \frac{1}{8}m(n), \varepsilon(n))$ -strong-two-block extractor from Theorem 41, where  $r_3(n) = O(\log(\frac{n}{\varepsilon}))$  and  $m'(n) \geq \frac{1-\gamma}{6} \cdot m(n)$ . Take  $X_5 = \text{EXT}_2(X'_4, X''_4, U_{r_3})$ , where  $U_{r_3}$  is a uniform random distribution on  $\{0, 1\}^{r_3(n)}$ , where  $(X'_4, X''_4) = X_4$ .

► **Lemma 43.**  $X_5$  is  $5\varepsilon(n)$  close to uniform.

The circuit depth of  $\text{EXT}_2$  is  $O(a + c + 1)^2$  by Theorem 41.

Now we prove the main theorem of this section:

► **Theorem 44.** For any constant  $a, c > 0, \gamma' \in (0, 1)$ , let  $k(n) \geq \frac{n}{\log^a(n)}, \varepsilon(n) \geq 2^{-\log^c(n)}$ . Then there exists a  $(k(n), \varepsilon'(n))$ -strong extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$ , such that  $r(n) = O(\log(\frac{n}{\varepsilon(n)}))$ , and  $m(n) \geq (1 - \gamma') \cdot k(n)$ .

Furthermore, the extractor can be implemented in  $\text{AC}^0$ , with  $O(a + c + 1)^2$  depth and  $\text{poly}(n)$  size.

**Proof.** The extractor  $\text{EXT}$  is defined as  $\text{EXT}(X_1, U_{r_1}, U_{r_2}, U_{r_3}) = X_5$ , where  $X_5$  is defined through the four steps above. Detailed analysis could be found in the full version. ◀

## 6 Extractors in $\text{NC}^1$

Our method can also construct extractors in  $\text{NC}^1$  with improved parameters. The construction consists of 3 parts:

1. Apply a condenser from [19]. It behaves like the GUV condenser but is computable in  $\text{NC}^1$ . It condenses the source into a source with a constant entropy rate. We regard the output as a block source.
2. For the second block, apply our error reduction method which outputs a seed of length  $O(\log^2 n \log(n/\varepsilon))$ .
3. Apply the improved Trevisan's extractor [26] to the first block, which outputs  $\Omega(k)$  bits of randomness.

The main theorem is as follows:

► **Theorem 45.** For every constant  $\gamma \in (0, 1)$  every  $k = k(n) \geq \Omega(\log^2(n)), \varepsilon = \varepsilon(n) \geq 2^{-O(\sqrt{k(n)})}$ , there exists a strong  $(k, \varepsilon)$  extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$  computable in  $\text{NC}^1$ , with  $r(n) = O(\log(n/\varepsilon))$ ,  $m(n) = (1 - \gamma)k(n)$ .

### 6.1 Condenser in $\text{NC}^1$

The first component in our construction is the condenser from [19]. A simplified version of their result is as follows:

► **Lemma 46** (condenser from [19]). For every  $k = k(n) \geq \Omega(\log^2(n)), \varepsilon = \varepsilon(n) \geq n \cdot 2^{-\sqrt{k(n)}/1024}$ , There exists  $m(n) \leq \frac{3}{2}k(n)$  and a function  $\text{Cond} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$  with  $r \leq 4 \log(\frac{n}{\varepsilon})$  such that  $\text{Cond}$  is a  $(k, k + r, \varepsilon)$ -condenser.

## 69:18 Randomness Extractors in $\text{AC}^0$ and $\text{NC}^1$ : Optimal up to Constant Factors

The condenser takes the input  $x$  as the representation of a degree  $\leq d = O(\frac{n}{\log q})$  polynomial over  $\mathbb{F}_q$  for some prime  $q \geq d, \log q \geq r$ . Denote the degree  $\leq d$  polynomial as  $f$ .

The condenser takes the seed  $y$  as a point in  $\mathbb{F}_q$ . Then the output is defined as:

$$\text{Cond}(x, y) = (y, f(y), f^{(1)}(y), \dots, f^{(s)}(y)) \quad (14)$$

for some  $s = s(n) \leq \frac{m(n)}{r(n)}$ .  $f^{(i)}$  denotes the  $i$ -th formal derivative of  $f$ .

To apply the condenser, we need to transform a source on  $\{0, 1\}^n$  to a source on  $\mathbb{F}_q$  and transform it back for the output. We use division to do the transformation, which is computable in  $\text{NC}^1$ .

The condenser itself requires two sorts of operations: polynomial evaluation and formal derivative. Denote  $f(x) = \sum_{i=0}^d a_i x^i$ . Then  $f^{(j)}(x) = \sum_{i=0}^d \frac{i!}{(i-j)!} a_i x^{i-j}$ . There are at most  $d^2$  such coefficients  $\frac{i!}{(i-j)!}$ , which can be precomputed and stored in the circuit. The multiplication of  $a_i$  and  $\frac{i!}{(i-j)!}$  can be done in  $\text{NC}^1$ . Therefore, the formal derivative is computable in  $\text{NC}^1$ .

The polynomial evaluation consists of three operations: calculating the powering  $x^{i-j}$ , multiplication and summation. The powering can be implemented with two steps:  $O(n)$ -th powering and division by  $q$ , which are computable in  $\text{NC}^1$  according to [4]. The multiplication and iterated summation are both in  $\text{NC}^1$ .

Putting it together, we can obtain the following lemma:

► **Lemma 47.** *The condenser from Lemma 46 is computable in  $\text{NC}^1$ .*

Regard the output of the condenser as  $(X_1, X_2)$ ,  $|X_1| = |X_2| = \frac{1}{2}m(n)$ . By Lemma 15,  $(X_1, X_2)$  is  $\varepsilon(n)$ -close to a  $(\frac{1}{2}m(n), \frac{1}{6}m(n), \frac{1}{2}m(n), \frac{1}{6}m(n))$ -source.

### 6.2 Error Reduction in $\text{NC}^1$

After condensing, we only need to handle an input  $(n, k)$  source  $X$  over  $\{0, 1\}^n$  with constant entropy rate  $\delta = \frac{k}{n}$ . To extract a seed of length  $O(\log n \log(n/\varepsilon))$ , we use almost the same procedure as in Section 4 despite some minor changes.

For the first step to convert the source to a somewhere source, we use the same extractors as in Section 4. We apply the extractors in parallel for  $t = \frac{\log n}{\log(1/\varepsilon)} = O(\sqrt{k})$  times. Then the output is  $\varepsilon$ -close to a somewhere  $(m_0, m_0 - \log(t))$ -source, where  $m_0 = \Omega(k)$ .

For the second step, we still apply the  $(m_0 - \log(t), 0.9m_0, \varepsilon)$ -merger from Corollary 26 to the output of the first step as in Section 4. Since  $\varepsilon \geq 2^{-O(\sqrt{k})}$  and  $t = \text{poly}(k)$ , the merger is computable in  $\text{NC}^1$ .

After applying the merger, we obtain a block-source with exponentially increasing length. We require a modification to Theorem 7 for the  $\text{NC}^1$  setting. The main difference is that the error is now  $2^{-O(\sqrt{k})}$  instead of  $2^{-\text{poly}(\log n)}$ . Also we setup the block length  $m_j = 3^j \cdot 10 \log \frac{n}{\varepsilon}$ ,  $j \in [l]$ , where  $l$  can still be  $O(\log n)$ , since  $\varepsilon = 2^{-O(\sqrt{k})}$ .

To extract from the block source, we require the following extractor in  $\text{NC}^1$ .

► **Lemma 48.** *For every constant  $\delta \in (0, 1]$  and every  $\varepsilon = 2^{-O(n)}$ , there exists an explicit  $(\delta n, \varepsilon)$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  in  $\text{NC}^1$ , where  $d = O(\log(n/\varepsilon))$ ,  $m = \Theta(\log(n/\varepsilon))$ .*

We use the sample-then-extract technique with leftover hash lemma to construct the extractor. Detailed analysis could be found in the full version.

Using the extractor to extract from the block source as in Section 4, we obtain a seed of length  $O(\log n \log(n/\varepsilon))$ .

One can use the iteration of Section 4 to stretch the output to  $O(\log^2 n \log(n/\varepsilon))$ .

This gives us the following lemma:

► **Lemma 49.** *For every  $\delta \in (0, 1)$ ,  $k = \delta n$ ,  $\varepsilon = \varepsilon(n) = 2^{-O(\sqrt{k})}$ , there exists a strong  $(k(n), \varepsilon(n))$ -extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$  computable in  $\text{NC}^1$ , with  $r(n) = O(\log(n/\varepsilon))$ ,  $m(n) = O(\log^2(n) \log(n/\varepsilon))$ .*

### 6.3 Improved Trevisan's Extractor in $\text{NC}^1$

With the seed of length  $O(\log^2 n \log(n/\varepsilon))$ , We apply the extractor from [26] to the first block of the block source. Their extractor could be implemented in  $\text{NC}^1$

► **Theorem 50 (Improved Trevisan's Extractor [26]).** *For every  $k = k(n)$ ,  $\varepsilon = \varepsilon(n)$ , there are explicit  $(k(n), \varepsilon(n))$ -extractors  $\text{EXT}_{\text{Trevisan}} : \{0, 1\}^n \times \{0, 1\}^{r(n)} \rightarrow \{0, 1\}^{m(n)}$  with  $r(n) = O(\log^2(n) \log(n/\varepsilon))$  and  $m(n) = \Omega(k(n))$ .*

*Moreover, the extractor  $\text{EXT}_{\text{Trevisan}}$  is computable in  $\text{NC}^1$ .*

### 6.4 Putting it together

Now we can prove Theorem 45.

**Proof of Theorem 45.** Take  $X$  as the input source. Let  $\text{Cond} : \{0, 1\}^n \times \{0, 1\}^{r_1(n)} \rightarrow \{0, 1\}^{m(n)}$  be the  $(k, k + r_1, \varepsilon/4)$ -condenser from Lemma 46. Take  $(X_1, X_2) = \text{Cond}(X, U_1)$ , where  $U_1$  is the seed of length  $r_1 = O(\log(n/\varepsilon))$ . By Lemma 15,  $(X_1, X_2)$  is  $\varepsilon/2$ -close to a  $(\frac{1}{2}m(n), \frac{1}{6}m(n), \frac{1}{2}m(n), \frac{1}{6}m(n))$ -source.

For  $X_2$ , apply the  $(\frac{1}{6}m(n), \varepsilon/4)$ -strong extractor  $\text{EXT}_1$  from Lemma 49 with seed  $U_2$  of length  $r_2 = O(\log(n/\varepsilon))$ . The output is  $Y = \text{EXT}_1(X_2, U_2)$  of length  $O(\log^2(n) \log(n/\varepsilon))$ .

For  $X_1$ , apply the  $(\frac{1}{2}m(n), \varepsilon/4)$ -extractor  $\text{EXT}_{\text{Trevisan}}$  from Theorem 50 with seed  $Y$ , which outputs a distribution  $W$  of length  $\Omega(k)$ .

By the property of  $\text{EXT}_1$ ,  $(X_1, Y)$  is  $3\varepsilon/4$ -close to  $(X_1, Y')$  such that  $Y'$  is a independent uniform distribution. Therefore  $W = \text{EXT}_{\text{Trevisan}}(X_1, Y)$  is  $\varepsilon$ -close to uniform.

The extractor  $\text{EXT}$  is defined as  $\text{EXT}(X, U_1, U_2) = W$ .  $\text{Cond}, \text{EXT}_1, \text{EXT}_{\text{Trevisan}}$  are all computable in  $\text{NC}^1$ . Therefore,  $\text{EXT}$  is computable in  $\text{NC}^1$ . ◀

## 7 Entropy lower bound for $\text{AC}^0$ dispersers

In the context of  $\text{AC}^0$  computation, not all sources are extractable. A well-known result of [10] shows that extracting even one bit of randomness is impossible for sources with entropy less than  $\frac{n}{\text{poly}(\log n)}$ . Similar result from [7] shows that extracting randomness with error less than  $2^{-\text{poly}(\log n)}$  is impossible for  $\text{AC}^0$  extractors.

In this section, we will extend the bound from extractors to dispersers. Dispersers are functions that take a source and a seed and output a distribution like extractors. The only difference is that the output distribution is not necessarily uniform, but rather supported on all but a small fraction of the codomain. We will show that strong  $\text{AC}^0$  dispersers for sources with entropy less than  $\frac{n}{\text{poly}(\log n)}$  do not exist.

► **Definition 51 (Disperser).** *A function  $\text{Disp} : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}^m$  is a  $(k, \varepsilon)$ -disperser if for every  $k$ -source  $X$  on  $\{0, 1\}^n$  and uniformly random variable  $Y$  on  $\{0, 1\}^r$ ,  $|\text{Supp}(\text{Disp}(X, Y))| \geq (1 - \varepsilon)2^m$ .*

*Furthermore,  $\text{Disp}$  is a strong  $(k, \varepsilon)$ -disperser if for every  $k$ -source  $X$  on  $\{0, 1\}^n$  and uniformly random variable  $Y$  on  $\{0, 1\}^r$ ,  $|\text{Supp}(Y, \text{Disp}(X, Y))| \geq (1 - \varepsilon)2^{r+m}$ .*

We remark that the requirement for  $X$  to have entropy  $\geq k$  can be replaced by a weaker requirement, which only requires  $\text{Supp}(X) \geq 2^k$ , without changing the definition.

Our proof is based on the new switching lemma for  $AC^0$  circuits by Rossman in [28]. Their original result says that every  $AC^0$  circuit can be reduced to a decision tree of arbitrary depth under a random restriction for all but a small fraction of the inputs. By restricting the inputs for the second time, it is reduced to a constant function.

► **Definition 52 (Restrictions).** A restriction  $\rho$  is a string on  $\{0, 1, *\}^n$ . We denote the application of  $\rho$  to  $x \in \{0, 1\}^n$  by  $\rho \circ x$ , which is defined as:

$$(\rho \circ x)_i = \begin{cases} \rho_i & \text{if } \rho_i \neq *, \\ x_i & \text{if } \rho_i = *. \end{cases} \quad (15)$$

The restriction on a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is defined as:

$$f|_\rho(x) = f(\rho \circ x). \quad (16)$$

We use  $R_p$  to denote the independent uniform random restriction with star probability  $p$ . That is, for every  $i \in [n]$ ,  $\Pr[R_p(i) = *] = p$ ,  $\Pr[R_p(i) = 0] = \Pr[R_p(i) = 1] = \frac{1-p}{2}$ .

The switching lemma for  $AC^0$  circuits is stated as follows:

► **Lemma 53 (Switching Lemma for  $AC^0$  circuits [28]).** For every  $\delta \in (0, 1)$ ,  $d > 0$ ,  $s = s(n)$ , there exists  $p = \frac{\delta}{\Theta(\log s)^{d-1}}$  such that for every  $AC^0$  circuit  $C$  of size  $s$  and depth  $d$ ,

$$\Pr_{\rho \sim R_p} [C|_\rho \text{ is not constant}] \leq \delta. \quad (17)$$

The following negative result for strong dispersers directly follows from the switching lemma.

► **Theorem 54.** For every  $d > 0$ ,  $s = s(n)$ , every constant  $\delta \in (0, 1)$ , if  $C : \{0, 1\}^n \times \{0, 1\}^r \rightarrow \{0, 1\}$  is a  $(k, \frac{1}{2} - \delta)$ -disperser that can be computed by a non-uniform  $AC$  circuit of size  $s$  and depth  $d$ , then  $k \geq \Theta(\frac{\delta n}{\log^{d-1} s})$ .

## 8 Open Questions

We mention the following open questions.

- For extractors in  $AC^0$ , can we further improve the circuit depth? The current depth is  $O(a + c + 1)^2$ . Is it possible to be linear in  $a + c + 1$ , while maintaining other parameters to be roughly the same?
- For extractors in  $NC^1$ , can we improve the plausible range of  $k$  and  $\varepsilon$ ? For example is it possible to give an  $NC^1$  construction that can work for all  $k, \varepsilon$ , matching the parameters in [13]?
- Some components of our  $NC^1$  computable extractors are actually in  $AC^0$ [2]. Is it possible to give an extractor in  $AC^0$ [2], with parameters optimal up to constant factors?
- For weak dispersers, we do not have a similar negative result to that of Section 7. The reason is that a single good seed in the seed space can make the disperser good enough, regardless of other seeds. So it remains an open question whether weak dispersers can be constructed in  $AC^0$ , specifically for sources with entropy less than  $\frac{n}{\text{poly}(\log n)}$ .



## References

- 1 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- 2 Z. Bar-Yossef, O. Reingold, R. Shaltiel, and L. Trevisan. Streaming computation of combinatorial objects. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 165–174, 2002.
- 3 David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5, 1986.
- 4 Paul Beame, Stephen A. Cook, and H. James Hoover. Log depth circuits for division and related problems. *SIAM J. Comput.*, 15(4):994–1003, 1986. doi:10.1137/0215070.
- 5 J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.
- 6 Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 125–136. IEEE, 2021. doi:10.1109/FOCS52979.2021.00021.
- 7 Kuan Cheng and Xin Li. Randomness extraction in ac0 and with small locality. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM) 2018*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
- 8 Gil Cohen, Dean Doron, Ori Sberlo, and Amnon Ta-Shma. Approximating iterated multiplication of stochastic matrices in small space. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 35–45. ACM, 2023. doi:10.1145/3564246.3585181.
- 9 Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the Method of Multiplicities, with Applications to Kakeya Sets and Mergers. *SIAM Journal on Computing*, 42(6):2305–2328, January 2013. doi:10/f5msx6.
- 10 Oded Goldreich, Emanuele Viola, and Avi Wigderson. On randomness extraction in AC0. In *Proceedings of the 30th Conference on Computational Complexity, CCC '15*, pages 601–668, Dagstuhl, DEU, June 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 11 Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties (preliminary version) a quality-size trade-off for hashing. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 574–584, 1994.
- 12 Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 440–449. ACM, 2007.
- 13 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM*, 56(4):1–34, June 2009. doi:10/ctbzhm.
- 14 Alexander Healy and Emanuele Viola. Constant-Depth circuits for arithmetic in finite fields of characteristic two. In *Proceedings of the 23rd Annual Conference on Theoretical Aspects of Computer Science, STACS'06*, pages 672–683, Berlin, Heidelberg, February 2006. Springer-Verlag. doi:10/df5dfs.
- 15 Alexander D Healy. Randomness-efficient sampling within nc1. *Computational Complexity*, 17(1):3–37, 2008.
- 16 Russel Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.
- 17 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 538–545. IEEE, 1995.
- 18 Russell Impagliazzo and Avi Wigderson. P=BPP unless E has sub-exponential circuits: Derandomizing the xor lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.

- 19 Itay Kalev and Amnon Ta-Shma. Unbalanced expanders from multiplicity codes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM) 2022*, volume 245 of *LIPICs*, pages 12:1–12:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.APPROX/RANDOM.2022.12.
- 20 Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. Extractors: Optimal up to Constant Factors. *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, June 2003. doi:10/bw2j9d.
- 21 Noam Nisan and Amnon Ta-Shma. Extracting randomness: A survey and new constructions. *Journal of Computer and System Sciences*, 58:148–173, 1999.
- 22 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.
- 23 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.
- 24 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors and depth-two superconcentrators. *Siam Journal on Discrete Mathematics*, 13:2–24, 2000.
- 25 Ran Raz, Omer Reingold, and Salil P. Vadhan. Error reduction for extractors. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99*, pages 191–201. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814591.
- 26 Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in trevisan’s extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002. doi:10.1006/JCSS.2002.1824.
- 27 Omer Reingold, Ronen Shaltiel, and Avi Wigderson. Extracting randomness via repeated condensing. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000*, pages 22–31. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892008.
- 28 Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of  $AC^0$ , 2017. URL: <https://users.cs.duke.edu/~br148/logsize.pdf>.
- 29 Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, 2002.
- 30 Ronen Shaltiel. An introduction to randomness extractors. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming*, 2011.
- 31 A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM Journal on Computing*, 28:1433–1459, 1999.
- 32 Amnon Ta-Shma. On extracting randomness from weak random sources. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 276–285, 1996.
- 33 Amnon Ta-Shma. Almost optimal dispersers. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, 1998*, pages 196–202. ACM, 1998. doi:10.1145/276698.276736.
- 34 Amnon Ta-Shma and Christopher Umans. Better condensers and new extractors from parvaresh-vardy codes. In *2012 IEEE 27th Conference on Computational Complexity*, pages 309–315. IEEE, 2012.
- 35 Roei Tell. Improved bounds for quantified derandomization of constant-depth circuits and polynomials. *computational complexity*, 28(2):259–343, 2019.
- 36 Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001. doi:10.1145/502090.502099.
- 37 Salil Vadhan. The unified theory of pseudorandomness. *SIGACT News*, 38, 2007.
- 38 Salil Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.
- 39 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *computational complexity*, 13(3-4):147–188, 2005.
- 40 Avi Wigderson and David Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999.
- 41 David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11(4):345–367, December 1997. doi:10/cr8kht.

# On Sampling from Ising Models with Spectral Constraints\*

Andreas Galanis ✉

University of Oxford, UK

Alkis Kalavasis ✉

Yale University, New Haven, CT, USA

Anthimos Vardis Kandiros ✉

Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

We consider the problem of sampling from the Ising model when the underlying interaction matrix has eigenvalues lying within an interval of length  $\gamma$ . Recent work in this setting has shown various algorithmic results that apply roughly when  $\gamma < 1$ , notably with nearly-linear running times based on the classical Glauber dynamics. However, the optimality of the range of  $\gamma$  was not clear since previous inapproximability results developed for the antiferromagnetic case (where the matrix has entries  $\leq 0$ ) apply only for  $\gamma > 2$ .

To this end, Kunisky (SODA'24) recently provided evidence that the problem becomes hard already when  $\gamma > 1$  based on the low-degree hardness for an inference problem on random matrices. Based on this, he conjectured that sampling from the Ising model in the same range of  $\gamma$  is NP-hard.

Here we confirm this conjecture, complementing in particular the known algorithmic results by showing NP-hardness results for approximately counting and sampling when  $\gamma > 1$ , with strong inapproximability guarantees; we also obtain a more refined hardness result for matrices where only a constant number of entries per row are allowed to be non-zero. The main observation in our reductions is that, for  $\gamma > 1$ , Glauber dynamics mixes slowly when the interactions are all positive (ferromagnetic) for the complete and random regular graphs, due to a bimodality in the underlying distribution. While ferromagnetic interactions typically preclude NP-hardness results, here we work around this by introducing in an appropriate way mild antiferromagnetism, keeping the spectrum roughly within the same range. This allows us to exploit the bimodality of the aforementioned graphs and show the target NP-hardness by adapting suitably previous inapproximability techniques developed for antiferromagnetic systems.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Problems, reductions and completeness; Theory of computation  $\rightarrow$  Random walks and Markov chains; Theory of computation  $\rightarrow$  Generating random combinatorial structures

**Keywords and phrases** Ising model, spectral constraints, Glauber dynamics, mean-field Ising, random regular graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.70

**Category** RANDOM

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.07645>

**Funding** Vardis Kandiros was supported by a Fellowship of the Eric and Wendy Schmidt Center at the Broad Institute of MIT and Harvard and by the Onassis Foundation-Scholarship ID: F ZP 016-1/2019-2020.

---

\* For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.



© Andreas Galanis, Alkis Kalavasis, and Anthimos Vardis Kandiros;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 70; pp. 70:1–70:14



Leibniz International Proceedings in Informatics

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

## 1 Introduction

The Ising model with a symmetric interaction matrix  $J \in \mathbb{R}^{N \times N}$  is a probability distribution  $\mu_J$  over  $\{-1, 1\}^N$  with

$$\mu_J(\sigma) = \frac{1}{Z_J} \exp\left(\frac{1}{2}\sigma^\top J\sigma\right) \text{ for all vectors } \sigma \in \{-1, 1\}^N,$$

where the normalizing constant  $Z_J = \sum_{\sigma \in \{-1, 1\}^N} \exp\left(\frac{1}{2}\sigma^\top J\sigma\right)$  is the *partition function* of the model. The most well-studied setting for the Ising model is when the underlying matrix  $J$  corresponds to the adjacency matrix of a graph  $G$ , scaled by a real parameter  $\beta$  which corresponds to the (inverse) temperature;<sup>1</sup> for  $\beta > 0$  the model is called ferromagnetic, and antiferromagnetic otherwise. The more general setting with non-uniform weights in the entries of  $J$  arises frequently in statistical learning settings, see, e.g., [9, 30, 25, 12].

The Ising model is the most fundamental example of a spin system, capturing how local interactions affect the global macroscopic behaviour, see [48, 41, 40, 33] for applications in various areas. From a computer science perspective, sampling from the Ising model plays a key role in various learning and inference problems. Understanding the limits of efficient sampling has therefore been a major focus in the literature, yielding new algorithmic techniques as well as exploring the power of classical algorithms (such as Glauber dynamics) and their connections to phase transitions in statistical mechanics; we briefly review some of the relevant literature below.

The prototypical setting where the problem of sampling for the Ising model has been studied is lattices (such as  $\mathbb{Z}^2$ ), where the landscape for Markov-chain algorithms has been well-understood [38, 39, 37]. Random graph models have also been considered more recently such as sparse random graphs [43, 13, 6, 16, 36] or the Sherrington-Kirkpatrick model [18, 17, 26]. More closely related to the setting considered in this paper is the case of general graphs. In the ferromagnetic case, where the entries of  $J$  are all nonnegative, the classical algorithm by Jerrum and Sinclair [28] gives a poly-time sampler (albeit with a relatively large running-time polynomial), see also [24, 19]. In the antiferromagnetic case, the problem is more interesting for bounded-degree graphs, where in the case of uniform weights the existence of polynomial-time algorithms is connected to the uniqueness threshold, see [46, 47, 21, 45, 35].

Recently, the development of spectral independence [4, 1] has given tight results on the performance of Glauber dynamics. This has led to nearly linear-time algorithms in various settings, see e.g., [11, 10, 18, 3, 31] and has made it possible to connect the performance of Glauber dynamics with the eigenvalues of the underlying matrix  $J$ . In this direction, [18, 3] show that Glauber dynamics is fast mixing when  $\lambda_{\max}(J) - \lambda_{\min}(J) < 1$  which significantly improves upon the standard Dobrushin's uniqueness condition (the latter only applies when  $\sum_j |J_{ij}| < 1$  for all  $i \in [N]$ ).

On the other side, the optimality of these algorithmic results in terms of the spectrum is less clear. It is known [34, 14] that Glauber dynamics mixes slowly in the complete graph for temperatures  $\beta > 1$ , which corresponds precisely to the condition  $\lambda_{\max}(J) - \lambda_{\min}(J) > 1$  by taking  $J$  to be the adjacency matrix of the  $N$ -vertex complete graph, scaled by  $\beta/N$ . This does not however translate in a straightforward way to hardness results and does not preclude the possibility that various alternative methods could potentially go beyond the 1-gap, see, e.g., [44, 31, 27] for some recent approaches using variational methods. To this

---

<sup>1</sup> Note that in this parametrization  $\mu_J(\sigma) \propto \exp(\frac{1}{2}\beta\sigma^\top A\sigma)$ , where  $A$  is the adjacency matrix of the graph.

end, Kunisky [32] gave further evidence that  $\lambda_{\max}(J) - \lambda_{\min}(J) > 1$  is hard for sampling via a reduction to hypothesis testing in a Wishart negatively-spiked matrix model that involves random matrices (which is known to resist low-degree algorithms [5]). Kunisky also posed the conjecture that in fact NP-hardness for sampling under spectral constraints should hold when  $\lambda_{\max}(J) - \lambda_{\min}(J) > 1$ . To add a bit to the mystery, it is noteworthy that the inapproximability results for the antiferromagnetic case (mentioned earlier) only apply roughly when  $\lambda_{\max}(J) - \lambda_{\min}(J) > 2$ , see also below for a more detailed discussion.

## 1.1 Our results

Our aim in this work is to address Kunisky’s conjecture and close the gap between algorithmic and NP-hardness results. In particular, we answer in the affirmative the conjecture in [32], obtaining NP-hardness results that complement the algorithmic results of [18, 3]. This completes the program initiated in [32], i.e., showing that Glauber is effectively optimal for “general-purpose” Ising model sampling, and clarifies the picture in terms of the computational complexity landscape under spectral constraints.

To formally state the result, we define the following computational problem.

*Problem:* SPECTRALISING( $\gamma$ )

*Input:* A symmetric matrix  $J \in \mathbb{R}^{N \times N}$ , with  $\lambda_{\max}(J) - \lambda_{\min}(J) < \gamma$ .

*Output:* The partition function  $Z_J = \sum_{\sigma \in \{-1, +1\}^N} \exp\left(\frac{1}{2} \sigma^\top J \sigma\right)$ .

► **Theorem 1.** *Fix any real  $\gamma > 1$ . Then, it is NP-hard to approximate SPECTRALISING( $\gamma$ ), even within an exponential factor  $2^{cN}$  for some constant  $c = c(\gamma) > 0$ .*

This confirms Conjecture 1.9 of [32] and complements the algorithm of [18, 3]. Using Theorem 1, we get the following result using the standard reduction [29] from counting to sampling (the problem is self-reducible under scaling of the matrix  $J$ ). Recall, the total variation distance between probability distributions  $\mu$  and  $\nu$  is defined as  $\text{TV}(\mu, \nu) = \frac{1}{2} \|\mu - \nu\|_1$ .

► **Corollary 2.** *For every real  $\gamma > 1$ , the following holds. Suppose there is a poly-time sampler that, on input a symmetric matrix  $J \in \mathbb{R}^{N \times N}$  with  $\lambda_{\max}(J) - \lambda_{\min}(J) < \gamma$  and  $\delta > 0$ , returns a configuration  $\sigma$  whose distribution is within TV distance  $\delta$  from  $\mu_J$ . Then NP = RP.*

As we will explain next, it is also possible to obtain a more refined version of Theorem 1, for the restricted case where each row of the interaction matrix  $J$  has at most  $d$  non-zero entries, for some fixed integer  $d \geq 4$ .

*Problem:* BOUNDED SPECTRALISING( $d, \gamma$ )

*Input:* A symmetric matrix  $J \in \mathbb{R}^{N \times N}$ , with  $\leq d$  non-zero entries per row and  $\lambda_{\max}(J) - \lambda_{\min}(J) < \gamma$ .

*Output:* The partition function  $Z_J = \sum_{\sigma \in \{-1, +1\}^N} \exp\left(\frac{1}{2} \sigma^\top J \sigma\right)$ .

► **Theorem 3.** *Fix any integer  $d \geq 4$  and real  $\gamma > \frac{1}{2} \ln\left(1 + \frac{2}{d-3}\right)(d-1 + 2\sqrt{d-2})$ . Then, it is NP-hard to approximate BOUNDED SPECTRALISING( $d, \gamma$ ), even within an exponential factor  $2^{cN}$  for some constant  $c = c(\gamma) > 0$ .*

Note that when taking the limit  $d \rightarrow \infty$  in the above bound, we recover the spectral condition  $\gamma > 1$  of Theorem 1, so asymptotically the bound is tight; we are not aware of algorithmic results that apply specifically to the  $d$ -sparse setting under the spectral condition.

We remark further that applying the results of [47, 21] would yield hardness only in the setting where  $\gamma > d \ln(1 + \frac{2}{d-2})$  (see [32, Section 1.2] for a detailed description on how to translate the results), so Theorem 3 improves on this by roughly a factor of 2 asymptotically. It should be noted however that the setting in these results is more restrictive (negative weights, which have the same value on all edges) and hence not directly comparable.

## 1.2 Techniques

Before giving the proofs, we explain briefly the main idea behind Theorem 1, the idea for Theorem 3 is almost identical, modulo the gadget used in the reduction.

The key ingredient in obtaining Theorem 1 is to exploit the slow mixing of Glauber dynamics on the complete graph in a suitable way. Recall that [34] showed exponential mixing time for Glauber dynamics on the  $N$ -vertex complete graph when the weights on the edges are ferromagnetic equal to  $\beta/N$  (entry-wise) for any  $\beta > 1$  (note that the corresponding matrix  $J$  has  $\lambda_{\max}(J) - \lambda_{\min}(J) = \beta$ ). Intuitively, the slow mixing is caused because the distribution exhibits bimodality, i.e., it is concentrated around two modes/“phases” corresponding roughly to the all-plus and all-minus configurations (see Section 2 for more details). Therefore, we would like to use the binary behaviour of the complete graph as a gadget in the reduction. The main trouble here is caused by the ferromagnetic interactions which cannot typically be related to NP-hard problems; by contrast, in the antiferromagnetic case  $\beta < 0$ , the max-probability configurations in the Ising distribution correspond to maximum cuts (when  $J$  encodes the adjacency matrix of a graph), and the respective gadgets in the constructions had bipartite structure.<sup>2</sup>

Hence, in order to get NP-hardness, we need to introduce some “mild” antiferromagnetism (small negative weights): mild to keep the spectrum unchanged and antiferromagnetic to allow us to reduce from an NP-hard problem (we will use MaxCut); this is quite different than the approach of [32] where the positive and negative entries in the constructed instance are more heavily mixed up (randomly). At this stage, the main observation is that the previous reductions used in the antiferromagnetic case [46, 47, 21] can accommodate this relatively easily; the only difference here is that we need to use small negative weights to connect disjoint copies of the gadgets, and amplify their effect using appropriately-sized matchings; conveniently, since the matchings (with the small weights on their edges) correspond to a low-rank perturbation, the spectrum of the underlying matrix is close to that of the complete graph.

The proof of Theorem 3 is almost identical. The main difference needed to make our construction sparse is to use a random  $d$ -regular graph as the gadget, which is known to exhibit slow mixing when  $\beta > \beta_d := \frac{1}{2} \ln(1 + \frac{2}{d-2})$  [22, 13, 43, 42], with a similar bimodal behaviour to that of the complete graph for  $\beta > 1$ . Relative to the spectrum, the well-known result of Friedman [20] shows that the adjacency matrix  $A$  of a random  $d$ -regular graph satisfies w.h.p.  $\lambda_{\max}(A) - \lambda_{\min}(A) \leq \lambda_d + \epsilon$  for any constant  $\epsilon > 0$ , where  $\lambda_d := d + 2\sqrt{d-1}$ . For technical reasons (see Remark 10 for details), we need to actually use a  $(d-1)$ -regular graph as a gadget in the reduction, so the argument sketched above yields NP-hardness when  $\gamma > \beta_{d-1} \lambda_{d-1}$  and  $d \geq 4$ .

---

<sup>2</sup> As a side note, we remark that the factor-2 gap from the antiferromagnetic setting (mentioned below Theorem 3) comes from the use of bipartite gadgets in these results, which have a symmetric spectrum around zero and hence effectively double the range of the eigenvalues.

### 1.3 Outline and Discussion

We give the details of the gadget in Section 2 and the reduction in Section 3.1. This gives a self-contained proof of Theorem 1; for Theorem 3 the argument is identical modulo the use of the (random)  $d$ -regular graph as the gadget, for which we need to import a couple of non-trivial results from the literature.

As a final remark before proceeding to the proofs, it would be interesting to explore whether the statistical hardness perspective from [32] (or some variant) perhaps applies to other counting/sampling problems where NP-hardness results are unlikely, such as approximating the number of independent sets in a bipartite graph [15], or approximating the partition function of the ferromagnetic Potts model [23]. Another related question is whether such statistical hardness results can be invoked on sparse random graph models where the spectral threshold  $\lambda_{\max}(J) - \lambda_{\min}(J) = 1$  (that applies to worst-case instances) is known not to be tight (see [10, 31, 36]).

## 2 The Gadget of Theorem 1

Our main gadget will be a clique graph  $K_n = (V, E)$  with  $n$  vertices, where  $V = \{1, 2, \dots, n\}$ . We will consider  $n$  to be an absolute (large) constant that we will choose later. For a small integer  $t > 0$ , let  $S \subseteq V$  be an arbitrary subset of  $V$  with  $|S| = t$ . Let  $r = n - t$ . Intuitively,  $S$  contains the nodes that will be used to connect the gadgets with each other.

We define the phase of the configuration  $\sigma \in \{-1, 1\}^n$  on  $V \setminus S$  as

$$Y_\sigma = \mathbf{1}\left\{\sum_{i \in V \setminus S} \sigma_i > 0\right\} - \mathbf{1}\left\{\sum_{i \in V \setminus S} \sigma_i \leq 0\right\}.$$

Note that the phase of a configuration is defined using only the spins in  $V \setminus S$ . For any fixed  $\beta > 0$ , consider solutions to the equation

$$\ln \frac{1 - \alpha}{\alpha} + 2\beta(2\alpha - 1) = 0 \tag{1}$$

for  $\alpha \in [0, 1]$ . It is not hard to see that for  $\beta > 1$  there are exactly three solutions  $\alpha = q^-, 1/2, q^+$  which satisfy  $q^+ - 1/2 = 1/2 - q^- > 0$ . Using these, we define the product measure  $Q_S^+$  (resp.  $Q_S^-$ ) on configurations on  $S$ , where each spin takes the value  $+1$  with probability  $q^+$ , and  $-1$  with probability  $1 - q^+$  (resp.  $q^-$  and  $1 - q^-$ ). Concretely, for  $\tau \in \{-1, +1\}^S$ , we have

$$Q_S^\pm(\tau) = (q^\pm)^{\frac{\sum_{i \in S} \tau_i + t}{2}} (1 - q^\pm)^{\frac{t - \sum_{i \in S} \tau_i}{2}} = (q^\pm(1 - q^\pm))^{t/2} \left(\frac{q^\pm}{1 - q^\pm}\right)^{\frac{\sum_{i \in S} \tau_i}{2}}. \tag{2}$$

We now state a lemma that presents the basic properties of the Ising model on our gadget graph. A similar lemma appears in the seminal results of [46, 47]. Informally, the lemma states that conditioned on the phase of the spins in  $V \setminus S$ , the spins in  $S$  behave almost *independently* from each other, with bias depending on the phase.

► **Lemma 4.** *Let  $\beta > 1$ . Then, for any real  $\epsilon > 0$  and integer  $t \geq 1$ , for all sufficiently large integers  $n = n(t, \epsilon)$  such that  $n - t$  is odd, the following holds for the Ising model with interaction matrix  $J \in \mathbb{R}^{n \times n}$  given by  $J = \frac{\beta}{n-t} \mathbf{1}\mathbf{1}^\top$ , where  $\mathbf{1}$  is the  $n$ -dimensional vector with all ones.*

Let  $S \subseteq [n]$  be a subset of the vertices with  $|S| = t$ . Then:

1. *The phases on  $V \setminus S$  appear with the same probability, i.e.,  $\Pr_{\sigma \sim \mu_J}[Y_\sigma = +] = \Pr_{\sigma \sim \mu_J}[Y_\sigma = -] = 1/2$ .*

## 70:6 On Sampling from Ising Models with Spectral Constraints

2. Conditioned on the phase, the joint distribution of the spins in  $S$  is approximately given by the product distribution  $Q_S^\pm$ , i.e.,

$$\text{for any } \tau \in \{-1, +1\}^S, \text{ it holds that } \Pr_{\sigma \sim \mu_J} [\sigma_S = \tau \mid Y_\sigma = \pm] = (1 \pm \epsilon) Q_S^\pm(\tau).$$

**Proof.** Let  $r = n - t$ . For  $r$  odd (as in the statement of the lemma), we have by symmetry that the phases appear with equal probability. So, we focus on proving the second item. For a vector  $x$  with entries  $+1$  or  $-1$ , we denote by  $|x|$  the sum of its entries.

Let  $\alpha \in [0, 1]$  be such that  $\alpha r$  is an integer. For a configuration  $\tau \in \{-1, +1\}^S$ , let  $Z^\alpha(\tau)$  be the contribution to the partition function of configurations  $\sigma$  with  $\alpha r$  spins from  $V \setminus S$  set to  $+1$ ,  $(1 - \alpha)r$  spins from  $V \setminus S$  set to  $-1$  and  $\sigma_S = \tau$ . Concretely,

$$Z^\alpha(\tau) = \sum_{\sigma \in \{-1, +1\}^V; \sigma_S = \tau, |\sigma_{V \setminus S}| = (2\alpha - 1)r} \exp\left(\frac{1}{2} \sigma^\top J \sigma\right).$$

The number of configurations  $\sigma$  with  $\sigma_S = \tau$  and exactly  $\alpha r$  of the spins in  $V \setminus S$  equal to  $1$  is  $\binom{r}{\alpha r}$ . Using that  $J = \frac{\beta}{r} \mathbf{1}\mathbf{1}^\top$ , for each such  $\sigma$ , we have  $\frac{1}{2} \sigma^\top J \sigma = \frac{\beta}{r} (|\sigma_{V \setminus S}| + |\tau|)^2 = \frac{\beta}{2r} ((2\alpha - 1)r + |\tau|)^2$ . So,

$$Z^\alpha(\tau) = \binom{r}{\alpha r} \exp\left(\frac{\beta}{2} (2\alpha - 1)^2 r + \beta(2\alpha - 1)|\tau| + \frac{\beta}{2r} |\tau|^2\right). \quad (3)$$

We use the well-known approximation of the binomial coefficient using Stirling's approximation. This yields, for any  $\alpha \in [0, 1]$ , that

$$\binom{r}{\alpha r} = \exp(rH(\alpha) + o(r)), \quad (4)$$

where  $H(\alpha) := -\alpha \ln \alpha - (1 - \alpha) \ln(1 - \alpha)$  is the binary entropy function. Asymptotically in  $r$ , we can also ignore the term  $\exp(\frac{\beta}{2r} |\tau|^2)$ , so we obtain that

$$Z^\alpha(\tau) = \exp(rf(\alpha) + o(r)) \text{ where } f(\alpha) := H(\alpha) + \frac{\beta}{2} (2\alpha - 1)^2. \quad (5)$$

The function  $f(\alpha)$  plays a key role since for large  $r$  it controls the asymptotic order of  $Z^\alpha(\tau)$ . The important point, as we will see below, is that the global maximum of  $f$  is attained for  $\alpha = q^\pm$ .

Indeed, we have

$$f'(\alpha) = -\ln(\alpha) + \ln(1 - \alpha) + 2\beta(2\alpha - 1)$$

and  $f''(\alpha) = -\frac{1}{\alpha(1-\alpha)} + 4\beta$ . Since  $f''$  has at most two zeros, we have that  $f'$  has at most three distinct zeros and hence  $f$  has at most three critical points. For  $\beta > 1$ , we have  $f'(1/2) = 0$  and  $f''(1/2) = -4 + 4\beta > 0$ , so  $f$  has a local minimum at  $\alpha = 1/2$ ; therefore, the maximum of  $f$  in the interval  $[0, 1]$  is attained at some point  $\alpha \neq 1/2$ . Using the symmetry of  $f$  around  $\alpha = 1/2$ , there must be at least two global maxima, one in the interval  $(0, 1/2)$  and  $(1/2, 1)$ . Since  $f$  has at most three critical points (and  $1/2$  is one of them), we conclude that there are exactly two critical points/maxima other than  $\alpha = 1/2$ , which must therefore be the values  $q^+, q^-$  as defined in (1).

We are now ready to establish the second item of the lemma. We will argue about the  $+$  phase, but the other phase is completely symmetric. Let  $\tau, \tau' \in \{-1, 1\}^S$  be two configurations of spins in  $S$ . We have that

$$\frac{\Pr[\sigma_S = \tau \mid Y(\sigma_{V \setminus S}) = +]}{\Pr[\sigma_S = \tau' \mid Y(\sigma_{V \setminus S}) = +]} = \frac{\sum_{\alpha > 1/2} Z^\alpha(\tau)}{\sum_{\alpha > 1/2} Z^\alpha(\tau')}. \quad (6)$$



We will show that the sums in the numerator and denominator are dominated by  $\alpha$  values that are close to  $q^+$ . First, note that since  $q^+$  is the unique global maximum of  $f(\alpha)$  in the interval  $[1/2, 1]$ , for any arbitrarily small constant  $\delta > 0$ , there is  $\eta > 0$  such that  $f(\alpha) \leq f(q^+) - 3\eta$  for all  $\alpha > 1/2$  with  $\alpha \notin [q^+ - \delta, q^+ + \delta]$ . We pick  $\delta > 0$  sufficiently small and  $r > 0$  sufficiently large so that  $\exp(4\beta t\delta + \beta \frac{t^2}{r}) < \epsilon/2$ . Since  $|\tau| \leq t$ , it follows that for  $r$  large enough it holds that

$$\sum_{\alpha > 1/2; |\alpha - q^+| > \delta} Z^\alpha(\tau) \leq \exp(r(f(q^+) - 2\eta)).$$

By the continuity of  $f$ , for  $\alpha = q^+ + O(1/r)$  we have  $f(\alpha) = f(q^+) + O(1/r)$  and therefore

$$\sum_{\alpha > 1/2; |\alpha - q^+| \leq \delta} Z^\alpha(\tau) \geq \exp(r(f(q^+) - \eta)).$$

It follows that

$$\frac{\sum_{\alpha > 1/2; |\alpha - q^+| > \delta} Z^\alpha(\tau)}{\sum_{\alpha > 1/2; |\alpha - q^+| \leq \delta} Z^\alpha(\tau)} \leq \exp(-\eta r) \leq \epsilon/2. \quad (7)$$

for all sufficiently large  $r$ . Thus,

$$\begin{aligned} \frac{\sum_{\alpha > 1/2} Z^\alpha(\tau)}{\sum_{\alpha > 1/2} Z^\alpha(\tau')} &\leq \frac{\sum_{\alpha > 1/2} Z^\alpha(\tau)}{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau')} = \frac{\sum_{\alpha > 1/2} Z^\alpha(\tau)}{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau)} \cdot \frac{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau)}{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau')} \\ &\leq (1 + \epsilon/2) \frac{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau)}{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau')}, \end{aligned} \quad (8)$$

where the last inequality follows from (7).

On the other hand, for any  $\alpha$  with  $|\alpha - q^+| \leq \delta$ , using (3) we get

$$\begin{aligned} \frac{Z^\alpha(\tau)}{Z^\alpha(\tau')} &= \exp\left(\beta(2\alpha - 1)(|\tau| - |\tau'|) + \frac{\beta(|\tau|^2 - |\tau'|^2)}{2r}\right) \\ &\leq \exp(4\beta t\delta + \beta \frac{t^2}{r}) \\ &\exp(\beta(2q^+ - 1)(|\tau| - |\tau'|)) \leq (1 + \epsilon/2) \exp(\beta(2q^+ - 1)(|\tau| - |\tau'|)), \end{aligned} \quad (9)$$

where the last inequality follows from the choice of  $\delta$  and  $r$ . Using the definition (2) and the fact that  $q^+$  is a solution of (1), i.e., that  $f'(q^+) = 0$ , we have that

$$\exp(\beta(2q^+ - 1)(|\tau| - |\tau'|)) = \left(\frac{q^+}{1 - q^+}\right)^{\frac{|\tau| - |\tau'|}{2}} = \frac{Q_S^+(\tau)}{Q_S^+(\tau')}.$$

Hence, from (9) we obtain that  $\frac{Z^\alpha(\tau)}{Z^\alpha(\tau')} \leq (1 + \epsilon/2) \frac{Q_S^+(\tau)}{Q_S^+(\tau')}$ . Since this holds for all  $\alpha$  with  $|\alpha - q^+| \leq \delta$ , we have

$$\frac{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau)}{\sum_{|\alpha - q^+| \leq \delta} Z^\alpha(\tau')} \leq (1 + \epsilon/2) \frac{Q_S^+(\tau)}{Q_S^+(\tau')}. \quad (10)$$

Combining this with (6) and (8), we obtain that

$$\frac{\Pr[\sigma_S = \tau | Y(\sigma_{V \setminus S}) = +]}{\Pr[\sigma_S = \tau' | Y(\sigma_{V \setminus S}) = +]} \leq (1 + \epsilon) \frac{Q_S^+(\tau)}{Q_S^+(\tau')}.$$

By interchanging the roles of  $\tau, \tau'$ , we also obtain the inverse inequality, so

$$(1 - \epsilon) \frac{Q_S^+(\tau)}{Q_S^+(\tau')} \leq \frac{\Pr[\sigma_S = \tau | Y(\sigma_{V \setminus S}) = +]}{\Pr[\sigma_S = \tau' | Y(\sigma_{V \setminus S}) = +]} \leq (1 + \epsilon) \frac{Q_S^+(\tau)}{Q_S^+(\tau')}. \quad (11)$$

For  $\tau \in \{-1, 1\}^S$ , observe that we can expand the ratio

$$\frac{\Pr[\sigma_S = \tau | Y = +]}{Q_S^+(\tau)} = \frac{\sum_{\tau'} Q_S^+(\tau') \Pr[\sigma_S = \tau | Y = +]}{\sum_{\tau'} Q_S^+(\tau) \Pr[\sigma_S = \tau' | Y = +]}$$

so using that  $\min_i \frac{a_i}{b_i} \leq \frac{\sum_i a_i}{\sum_i b_i} \leq \max_i \frac{a_i}{b_i}$  for non-negative  $(a_i)_i, (b_i)_i$ , we obtain from (11) that

$$\left| \frac{\Pr[\sigma_S = \tau | Y = +]}{Q_S^+(\tau)} - 1 \right| \leq \max_{\tau'} \left| \frac{Q_S^+(\tau') \Pr[\sigma_S = \tau | Y = +]}{Q_S^+(\tau) \Pr[\sigma_S = \tau' | Y = +]} - 1 \right| \leq \epsilon.$$

This finishes the proof.  $\blacktriangleleft$

### 3 Proofs of Main Results

#### 3.1 Proof of Theorem 1

Let  $\gamma > 1$  and  $\beta = (1 + \gamma)/2 > 1$ . Following the technique in [46, 47, 21], we reduce MAXCUT on 3-regular graphs to SPECTRALISING( $\gamma$ ).

Consider a 3-regular graph  $H = (V_H, E_H)$  with  $|V_H| = m$  vertices, an instance of MAXCUT. Let  $G$  be the clique graph on  $n$  vertices, with a subset  $S$  of the vertices with  $|S| = t$  that will be used as terminals (cf. Lemma 4); for convenience, we assume that  $t > 0$  is a multiple of 3 (with  $n \gg 3t$ ). We construct an instance  $H^G$  of SPECTRALISING( $\gamma$ ) as follows:

- We replace each node  $v \in V_H$  with a distinct copy of the gadget clique graph  $G$ . In particular, for any  $v \in V_H$ , consider a copy  $G_v = (W_v, E_v)$  of the gadget  $G$ ; each edge in  $E_v$  has weight  $w_+ = \beta/r$  as in Lemma 4, where recall that  $\beta = (1 + \gamma)/2 > 1$ . For each  $v \in V_H$ , let  $S_v \subseteq W_v$  be a subset of the vertices in  $G_v$  of size  $t = n - r$ . Let  $\hat{H}^G$  be the disjoint union of the  $G_v$ 's for  $v \in H$ . Note that the number of vertices of  $\hat{H}^G$  is  $nm$ .
- We now describe how to encode the edges of  $H$  using connections between the gadgets (which will complete the construction of  $H^G$ ). Assume that the node  $u \in V_H$  has neighbors  $v_1, v_2, v_3$  in  $H$ , i.e.,  $(u, v_i) \in E_H, i = 1, \dots, 3$ . Then, we partition  $S_u$  into subsets  $S_u^i$  of size  $t/3$  each. Each subset  $S_u^i$  corresponds to one of the three neighbors of  $u$ . Then, for each  $i = 1, 2, 3$ , we add a perfect matching between  $S_u^i$  and the corresponding subset  $S_{v_i}^j$  of  $S_{v_i}$  that corresponds to  $u$ . The weight of each of these edges in the matching will be  $w_- = (1 - \gamma)/5 < 0$ , since  $\gamma > 1$ . This antiferromagnetic structure across different copies will be crucial in order to approximate  $\maxcut(H)$  by approximating the partition function of  $H^G$ .

Let  $J$  be the adjacency matrix of the weighted graph  $H^G$ . We first show that the spectrum of  $J$  has the desired properties, i.e., that  $\lambda_{\max}(J) - \lambda_{\min}(J) < \gamma$ .

▷ **Claim 5 (Structure of  $H^G$ ).** The symmetric matrix  $J = D + E \in \mathbb{R}^{nm \times nm}$ , where  $D$  is a block diagonal matrix where the matrix of each block of size  $n \times n$  is  $\frac{\beta}{r} \mathbf{1}\mathbf{1}^\top$  and  $E$  contains in each row exactly one non-zero element of magnitude  $(1 - \gamma)/5$ .

Proof. By construction since  $n$  is the number of vertices of the gadget and  $m$  is the number of vertices of the input graph.  $\blacktriangleleft$

▷ **Claim 6 (Spectrum Preservation).** For any integer  $t > 0$ , there exists  $n(t, \gamma) > 0$ , such that for  $n > n(t, \gamma)$  it holds  $|\lambda_{\max}(J) - \lambda_{\min}(J)| < \gamma$ .

*Proof.* We will use Claim 5. Using Weyl's inequality (see Chapter 3 in [7]), which controls the eigenspectrum of a matrix under small perturbations of the entries, we have that for any  $i$ , it holds that  $|\lambda_i(J) - \lambda_i(D)| \leq \|E\|$ , where  $\|E\|$  is the spectral norm of  $E$ . By definition,  $E$  has one element in each row of absolute value  $(\gamma - 1)/5$ , so  $\|E\| \leq \frac{\gamma-1}{5}$ . It follows that

$$\begin{aligned} |\lambda_{\max}(J) - \lambda_{\min}(J)| &\leq |\lambda_{\max}(J) - \lambda_{\max}(D)| + |\lambda_{\max}(D) - \lambda_{\min}(D)| + |\lambda_{\min}(D) - \lambda_{\min}(J)| \\ &\leq \frac{2(\gamma-1)}{5} + \frac{n}{r} \frac{1+\gamma}{2}. \end{aligned} \tag{12}$$

In the above we used the well-known fact that the spectrum of  $D$  is the spectrum of each of the blocks, which, in turn, is equal to

$$\lambda_{\max}(D) - \lambda_{\min}(D) = \frac{n}{r} \frac{1+\gamma}{2},$$

since each block is a rank-1 matrix. Now, recall that  $n = r + t$ , so by choosing  $r$  sufficiently large we can make  $\frac{n}{r} < \frac{6\gamma+4}{5\gamma+5}$ , which implies that the right hand side in (12) is  $< \gamma$ . ◁

We next show that if we could approximate  $Z_J$  within an arbitrarily small exponential factor in poly-time, we would obtain a PTAS for  $\text{maxcut}(H)$ . This part of the argument is largely based on the techniques of [47]; we first state the following lemma whose proof is given for completeness in the full version.

► **Lemma 7.** *It holds that*

$$(1 - 4\epsilon)^m 2^{-m} \leq \frac{Z_{H^G} / Z_{\hat{H}^G}}{A^{3mt/2} (B/A)^{\text{maxcut}(H)t/3}} \leq (1 + 4\epsilon)^m,$$

where  $B > A > 0$  are constants depending only on  $\gamma$ .

With these pieces at hand, we are now ready to complete the reduction for Theorem 1, which we restate here for convenience.

► **Theorem 1.** Fix any real  $\gamma > 1$ . Then, it is NP-hard to approximate  $\text{SPECTRALISING}(\gamma)$ , even within an exponential factor  $2^{cN}$  for some constant  $c = c(\gamma) > 0$ .

**Proof.** Assume that for any arbitrarily small constant  $\delta > 0$ , there is an oracle  $\text{approx}_\delta$  such that, for any  $J$  with  $\lambda_{\max}(J) - \lambda_{\min}(J) \leq \gamma$ , we have that, when  $F = \text{approx}_\delta(J)$ ,  $|F - \log(Z(J))| \leq \delta m$ . We will show how to obtain a PTAS for MAXCUT on 3-regular graphs, i.e., approximate MAXCUT on 3-regular graphs within an arbitrarily small factor.

Let  $H$  be a 3-regular graph  $H$  on  $m$  vertices, an instance of MAXCUT. The maximum cut of  $H$  is at least the expected value of a random cut which is equal to  $3m/4$ . We then construct  $H^G$  and  $\hat{H}^G$  as above. Observe that  $Z(\hat{H}^G)$  can be computed in poly-time since  $\hat{H}^G$  is a disjoint collection of constant-size gadget graphs. Moreover, by Claim 6,  $H^G$  is an instance of  $\text{SPECTRALISING}(\gamma)$ . So, we can use the oracle  $\text{approx}_\delta$  on  $H^G$ , which will give us an output  $F_H$  with the guarantee

$$|F_H - \log Z_{H^G}| \leq \delta mn.$$

Lemma 7 implies that

$$\frac{3 \log \left( \frac{Z_{H^G} / Z_{\hat{H}^G}}{A^{3mt/2} (1+4\epsilon)^m} \right)}{t \log(B/A)} \leq \text{maxcut}(H) \leq \frac{3 \log \left( \frac{2^m Z_{H^G} / Z_{\hat{H}^G}}{A^{3mt/2} (1-4\epsilon)^m} \right)}{t \log(B/A)}.$$

Thus, by using the output  $F_H$  we can compute upper and lower bounds for the maximum cut value, which differ by  $O((\delta n + 1)m/t)$ . Since  $m \leq 4/3 \max\text{cut}(H)$ , to show the desired PTAS for MAXCUT, it only remains to show that the quantity  $R = (\delta n + 1)/t$  can be made arbitrarily small, say less than some target value  $\zeta$ , where  $\zeta > 0$  is an arbitrary constant. We first take  $t$  to be sufficiently large, so that  $1/t < \zeta/2$  is sufficiently small. This makes  $n$  to be large, but still a constant, and hence  $n/t$  is a constant. So, by taking  $\delta$  small enough, we will have  $\delta n/t < \zeta/2$ , making  $R < \zeta$  as desired.

This yields the desired PTAS. Since MAXCUT is APX-hard [2], we conclude that it is NP-hard to approximate  $Z_J$  within some exponential factor, as wanted. ◀

### 3.2 Proof of Theorem 3

For integers  $d, n \geq 3$  with  $dn$  even, let  $G_{n,d}$  be a  $d$ -regular graph chosen uniformly at random among all such graphs with vertex set  $V = \{1, 2, \dots, n\}$ . Let  $S \subseteq [n]$  be an arbitrary subset of the vertices of size  $t$ . Consider the Ising distribution  $\mu_J$  with  $J = \beta A$  where  $A$  is the adjacency matrix of  $G$  and  $\beta > \frac{1}{2} \ln(1 + \frac{2}{d-2})$ .

The range of  $\beta$  corresponds to the so-called non-uniqueness regime on the  $d$ -regular tree; roughly, this implies that on the  $d$ -regular tree of height  $h$ , when we condition the leaves to be  $+$  and take the limit  $h \rightarrow \infty$ , the marginal probability that the root is plus converges to some value  $q^+ > 1/2$ . Similarly, when we condition the leaves to be  $-$ , the marginal probability that the root is plus converges to some value  $q^- < 1/2$ .<sup>3</sup>

It is well-known by now [13, 42] that this behaviour on the tree manifests itself on the random  $d$ -regular graph, roughly because of the tree-like neighborhoods in the latter. To make this more precise in our setting, analogously to Section 2, for a subset  $S \subseteq V$ , define the phase  $Y_S(\sigma)$  of a configuration  $\sigma \in \{-1, +1\}^V$  to be  $+$  if  $\sum_{i \in V \setminus S} \sigma_i \geq 0$ , and  $-$  otherwise. We also define the product measures  $Q_S^\pm$  on  $S$  analogously to (2), using now the values of  $q^+, q^-$  as defined above (see also Footnote 3). Then, the following lemma captures the main properties of the gadget that we need.

► **Lemma 8.** *Let  $d \geq 3$  be an integer and  $\beta > \frac{1}{2} \ln(1 + \frac{2}{d-2})$ . Then, for any real  $\epsilon > 0$  and integer  $t \geq 1$ , for all sufficiently large integers  $n = n(t, \epsilon)$  with  $n - t$  odd, the following holds with probability  $1 - \epsilon$  over the choice of  $G \sim G_{n,d}$ . Let  $S \subseteq V$  be a subset of vertices with  $|S| = t$ .*

*Consider the Ising model with interaction matrix  $J = \beta A$  where  $A$  is the adjacency matrix of  $G$ . Then:*

1.  $\lambda_{\max}(J) - \lambda_{\min}(J) \leq \beta(d + 2\sqrt{d-1}) + \epsilon$ .
2. The phases appear with the same probability, i.e.,  $\Pr_{\sigma \sim \mu_J}[Y_\sigma = +] = \Pr_{\sigma \sim \mu_J}[Y_\sigma = -] = 1/2$ .
3. Conditioned on the phase, the joint distribution of the spins in  $S$  is approximately given by the product distribution  $Q_S^\pm$ , i.e.,

$$\text{for any } \tau \in \{-1, +1\}^S, \text{ it holds that } \Pr_{\sigma \sim \mu_J}[\sigma_S = \tau \mid Y_\sigma = \pm] = (1 \pm \epsilon)Q_S^\pm(\tau).$$

**Proof.** The first item is Friedman's result [20], see also [8]. The second item is by symmetry of the configuration space (since  $n$  is odd). The third item follows by [42, Theorem 2.4], see also [13, Theorem 2.7] and [47, Proposition 4.2] for related results. Technically, there is a bit of work to translate the results here, we give the details in the full version. ◀

<sup>3</sup> To define  $q^+, q^-$  more explicitly, for  $\beta > \frac{1}{2} \ln(1 + \frac{2}{d-2})$ , let  $\tilde{q}^+ > 1 > \tilde{q}^- > 0$  be the solutions of  $x = \left(\frac{\exp(2\beta)x+1}{x+\exp(2\beta)}\right)^{d-1}$ . Then,  $q^+, q^-$  are defined from  $\frac{q^+}{1-q^+} = \tilde{q}^+ \frac{\exp(2\beta)\tilde{q}^++1}{\tilde{q}^++\exp(2\beta)}$  and  $\frac{q^-}{1-q^-} = \tilde{q}^- \frac{\exp(2\beta)\tilde{q}^-+1}{\tilde{q}^-+\exp(2\beta)}$ , see also [21, Section 3].

► **Remark 9.** We use the gadget of Lemma 8 for some large but otherwise constant value of  $n$ . So, we can find a  $d$ -regular graph  $G$  satisfying Items 1-3 of Lemma 8 in deterministic time.

We are now ready to prove Theorem 3, which we restate here for convenience.

► **Theorem 3.** Fix any integer  $d \geq 4$  and real  $\gamma > \frac{1}{2} \ln(1 + \frac{2}{d-3})(d-1 + 2\sqrt{d-2})$ . Then, it is NP-hard to approximate  $\text{BOUNDEDSPECTRALISING}(d, \gamma)$ , even within an exponential factor  $2^{cN}$  for some constant  $c = c(\gamma) > 0$ .

**Proof.** Let

$$\beta_{d-1} := \frac{1}{2} \ln\left(1 + \frac{2}{d-3}\right), \quad \lambda_{d-1} := d-1 + 2\sqrt{d-2} \tag{13}$$

and set  $\beta = \beta_{d-1} + \eta$ ,  $\lambda = \lambda_{d-1} + \eta$  where  $\eta > 0$  is a small constant so that  $\beta\lambda + 2\eta < \gamma$  (note that such an  $\eta$  exists since  $\gamma > \beta_{d-1}\lambda_{d-1}$ ).

Assume that we are given a 3-regular instance  $H$  of MAXCUT with  $m$  vertices. Let  $G$  be a  $(d-1)$ -regular gadget with  $n$  vertices for some sufficiently large  $n$ , i.e.,  $G$  satisfies Items 1-3 of Lemma 8 for degree  $d-1$  and  $\beta = \beta_{d-1} + \eta$ , see also Remark 9. So, according to Item 1 there, the interaction matrix  $J_G$  corresponding to  $G$  satisfies  $\lambda_{\max}(J_G) - \lambda_{\min}(J_G) \leq \beta\lambda$ .

Using  $G$ , the construction of the graph  $H^G$  is identical to that of Section 3.1, i.e., we have a distinct copy of  $G$  for each node of  $H$  and, for each pair of neighbouring nodes of  $H$ , we add a matching of size  $t/3$  between the corresponding gadgets using the vertices in  $S$ . Note that  $H^G$  has maximum degree  $d$ , so the interaction matrix of  $H^G$ , denoted by  $J$  henceforth, has at most  $d$  non-zero entries per row.

The weight of an edge inside the gadget is  $w_+ = \beta > 0$  and the weight of the edges that connect two gadgets is  $w_- = -\eta < 0$  (antiferromagnetic connections). Analogously to Claim 5, the symmetric matrix  $J$  can be written as  $D + E \in \mathbb{R}^{nm \times nm}$ , where (i)  $D$  is a block diagonal matrix with the matrix in each block being the  $n \times n$  adjacency matrix of  $G$  scaled by  $w_+$ , and (ii)  $E$  contains in each row exactly one non-zero element of magnitude  $w_-$ . The same argument as in the proof of Claim 6 gives that

$$\begin{aligned} |\lambda_{\max}(J) - \lambda_{\min}(J)| &\leq |\lambda_{\max}(J) - \lambda_{\max}(D)| + |\lambda_{\max}(D) - \lambda_{\min}(D)| + |\lambda_{\min}(D) - \lambda_{\min}(J)| \\ &\leq 2\eta + \beta\lambda < \gamma. \end{aligned}$$

This establishes that  $H^G$  is a valid instance of  $\text{BOUNDEDSPECTRALISING}(d, \gamma)$ .

Now, using Item 3 of Lemma 8, we obtain the exact same estimate as in Lemma 7 (with the same expressions for the constants  $A, B$  modulo the new values of  $w_+$  and  $w_-$ ), and therefore the same argument used in the proof of Theorem 1 applies verbatim to show NP-hardness of approximating the partition function within an arbitrarily small exponential factor. ◀

► **Remark 10.** Note that we could make the graph  $H^G$  to be  $d$ -regular for any integer  $d \geq 3$  by taking the gadget  $G$  to be a random  $d$ -regular graph with a matching of size  $t$  removed (and using the endpoints of the matching as the set  $S$  of terminals); this more refined construction has been used for example in the hardness results of [46, 47, 21]. While one can show the analogue of Items 2 and 3 with minor modifications (analogously to what was done in the proof of Lemma 8), the proof of Item 1 for this modified gadget seems to require more careful adaptation of the proofs in [20, 8]. It is nevertheless reasonable to expect that the same bound on the range of the eigenvalues as stated currently in Item 1 will still apply; provided this is indeed the case, one can improve slightly the parameters of Theorem 3 to  $d \geq 3$  and  $\gamma > \beta_d \lambda_d$ , where  $\beta_d, \lambda_d$  are as in (13).

## References

- 1 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1198–1211, 2020.
- 2 Paola Alimonti and Viggo Kann. Hardness of approximating problems on cubic graphs. In *Proceedings of the Third Italian Conference on Algorithms and Complexity*, CIAC '97, pages 288–298, 1997.
- 3 Nima Anari, Vishesh Jain, Frederic Koehler, Huy Tuan Pham, and Thuy-Duong Vuong. Entropic independence in high-dimensional expanders: Modified log-sobolev inequalities for fractionally log-concave polynomials and the ising model. *arXiv preprint*, 10:32–42, 2021. [arXiv:2106.04105](https://arxiv.org/abs/2106.04105).
- 4 Nima Anari, Kuikui Liu, and Shayan Oveis Gharan. Spectral independence in high-dimensional expanders and applications to the hardcore model. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1319–1330, 2020.
- 5 Afonso S Bandeira, Dmitriy Kunisky, and Alexander S Wein. Computational hardness of certifying bounds on constrained pca problems. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, page 78. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 6 Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, and Daniel Štefankovič. Fast sampling via spectral independence beyond bounded-degree graphs. *ACM Trans. Algorithms*, 20(1), January 2024.
- 7 Rajendra Bhatia. *Perturbation bounds for matrix eigenvalues*. SIAM, 2007.
- 8 Charles Bordenave. A new proof of friedman’s second eigenvalue theorem and its extension to random lifts. In *Annales Scientifiques de l’École Normale Supérieure*, volume 4, pages 1393–1439, 2020.
- 9 Guy Bresler. Efficiently learning Ising models on arbitrary graphs. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 771–782, 2015.
- 10 Yuansi Chen and Ronen Eldan. Localization schemes: A framework for proving mixing bounds for markov chains. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–122. IEEE, 2022.
- 11 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 2021, pages 1537–1550, 2021.
- 12 Yuval Dagan, Constantinos Daskalakis, Nishanth Dikkala, and Anthimos Vardis Kandiros. Learning Ising models from one or multiple samples. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 161–168, 2021.
- 13 Amir Dembo and Andrea Montanari. Ising models on locally tree-like graphs. *The Annals of Applied Probability*, 20(2):565–592, 2010.
- 14 Jian Ding, Eyal Lubetzky, and Yuval Peres. The mixing time evolution of glauber dynamics for the mean-field ising model. *Communications in Mathematical Physics*, 289(2):725–764, 2009.
- 15 Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38:471–500, 2004.
- 16 Charilaos Efthymiou and Kostas Zampetakis. On sampling diluted spin glasses using glauber dynamics. *arXiv preprint*, 2024. [arXiv:2403.08921](https://arxiv.org/abs/2403.08921).
- 17 Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Sampling from the sherrington-kirkpatrick gibbs measure via algorithmic stochastic localization. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 323–334. IEEE, 2022.
- 18 Ronen Eldan, Frederic Koehler, and Ofer Zeitouni. A spectral condition for spectral gap: fast mixing in high-temperature ising models. *Probability theory and related fields*, 182(3):1035–1051, 2022.

- 19 Weiming Feng, Heng Guo, and Jiaheng Wang. Swendsen-Wang dynamics for the ferromagnetic Ising model with external fields. *Information and Computation*, 294:105066, 2023.
- 20 Joel Friedman. *A proof of Alon's second eigenvalue conjecture and related problems*. American Mathematical Soc., 2008.
- 21 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.
- 22 Antoine Gerschenfeld and Andrea Montanari. Reconstruction for models on random graphs. In *2007 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 194–204, 2007.
- 23 Leslie Ann Goldberg and Mark Jerrum. Approximating the partition function of the ferromagnetic Potts model. *J. ACM*, 59(5), 2012.
- 24 Heng Guo and Mark Jerrum. Random cluster dynamics for the Ising model is rapidly mixing. *The Annals of Applied Probability*, 28(2):1292–1313, 2018.
- 25 Linus Hamilton, Frederic Koehler, and Ankur Moitra. Information theoretic properties of Markov random fields, and their algorithmic applications. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- 26 Brice Huang, Andrea Montanari, and Huy Tuan Pham. Sampling from spherical spin glasses in total variation via algorithmic stochastic localization. *arXiv preprint*, 2024. [arXiv:2404.15651](https://arxiv.org/abs/2404.15651).
- 27 Vishesh Jain, Frederic Koehler, and Andrej Risteski. Mean-field approximation, convex hierarchies, and the optimality of correlation rounding: a unified perspective. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1226–1236, 2019.
- 28 Mark Jerrum and Alistair Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on computing*, 22(5):1087–1116, 1993.
- 29 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.
- 30 Adam Klivans and Raghu Meka. Learning graphical models using multiplicative weights. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 343–354, 2017.
- 31 Frederic Koehler, Holden Lee, and Andrej Risteski. Sampling approximately low-rank Ising models: MCMC meets variational methods. In *Conference on Learning Theory*, pages 4945–4988. PMLR, 2022.
- 32 Dmitriy Kunisky. Optimality of Glauber dynamics for general-purpose Ising model sampling and free energy approximation. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5013–5028, 2024.
- 33 Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.
- 34 David A Levin, Malwina J Luczak, and Yuval Peres. Glauber dynamics for the mean-field Ising model: cut-off, critical power law, and metastability. *Probability Theory and Related Fields*, 146:223–265, 2010.
- 35 Liang Li, Pinyan Lu, and Yitong Yin. Correlation decay up to uniqueness in spin systems. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 67–84, 2013.
- 36 Kuikui Liu, Sidhanth Mohanty, Amit Rajaraman, and David X Wu. Fast mixing in sparse random Ising models. *arXiv preprint*, 2024. [arXiv:2405.06616](https://arxiv.org/abs/2405.06616).
- 37 Eyal Lubetzky and Allan Sly. Critical Ising on the square lattice mixes in polynomial time. *Communications in Mathematical Physics*, 313(3):815–836, 2012.
- 38 Fabio Martinelli and Enzo Olivieri. Approach to equilibrium of Glauber dynamics in the one phase region: I. The attractive case. *Communications in Mathematical Physics*, 161(3):447–486, 1994.

- 39 Fabio Martinelli and Enzo Olivieri. Approach to equilibrium of Glauber dynamics in the one phase region: II. The general case. *Communications in Mathematical Physics*, 161(3):487–514, 1994.
- 40 Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.
- 41 Marc Mézard, Giorgio Parisi, and Miguel Angel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.
- 42 Andrea Montanari, Elchanan Mossel, and Allan Sly. The weak limit of Ising models on locally tree-like graphs. *Probability Theory and Related Fields*, 152:31–51, 2012.
- 43 Elchanan Mossel and Allan Sly. Exact thresholds for Ising–Gibbs samplers on general graphs. *The Annals of Probability*, 41(1), 2013.
- 44 Andrej Risteski. How to calculate partition functions using convex programming hierarchies: provable bounds for variational methods. In *Conference on Learning Theory*, pages 1402–1416. PMLR, 2016.
- 45 Alistair Sinclair, Piyush Srivastava, and Marc Thurley. Approximation algorithms for two-state anti-ferromagnetic spin systems on bounded degree graphs. *Journal of Statistical Physics*, 155(4):666–686, 2014.
- 46 Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 287–296. IEEE, 2010.
- 47 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.
- 48 Michel Talagrand. *Mean field models for spin glasses: Volume I: Basic examples*, volume 54. Springer Science & Business Media, 2010.



# Approximate Degree Composition for Recursive Functions

Sourav Chakraborty ✉ 🏠 


Indian Statistical Institute, Kolkata, India

Chandrima Kayal ✉

Indian Statistical Institute, Kolkata, India

Rajat Mittal ✉ 🏠

Indian Institute of Technology Kanpur, India

Manaswi Paraashar ✉ 🏠 

University of Copenhagen, Denmark

Nitin Saurabh ✉ 🏠

Indian Institute of Technology Hyderabad, India

---

## Abstract

---

Determining the approximate degree composition for Boolean functions remains a significant unsolved problem in Boolean function complexity. In recent decades, researchers have concentrated on proving that approximate degree composes for special types of inner and outer functions. An important and extensively studied class of functions are the recursive functions, i.e. functions obtained by composing a base function with itself a number of times. Let  $h^d$  denote the standard  $d$ -fold composition of the base function  $h$ . The main result of this work is to show that the approximate degree composes if either of the following conditions holds:

- The outer function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is a recursive function of the form  $h^d$ , with  $h$  being any base function and  $d = \Omega(\log \log n)$ .
- The inner function is a recursive function of the form  $h^d$ , with  $h$  being any constant arity base function (other than AND and OR) and  $d = \Omega(\log \log n)$ , where  $n$  is the arity of the outer function.

In terms of proof techniques, we first observe that the lower bound for composition can be obtained by introducing majority in between the inner and the outer functions. We then show that majority can be *efficiently eliminated* if the inner or outer function is a recursive function.

**2012 ACM Subject Classification** Theory of computation

**Keywords and phrases** Approximate degree, Boolean function, Composition theorem

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.71

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.08385> [16]

**Funding** *Sourav Chakraborty*: supported by the Science & Engineering Research Board of the DST, India, through the MATRICS grant MTR/2021/000318.

*Manaswi Paraashar*: supported by ERC grant (QInteract, Grant Agreement No 101078107).

*Nitin Saurabh*: supported by the seed grant (SG/IITH/F285/2022-23/SG-123) from IIT Hyderabad.

## 1 Introduction

Representations of Boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  in terms of multivariate polynomials  $p(x)$  play a pivotal role in theoretical computer science. There are different notions of representations;



© Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, and Nitin Saurabh; licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 71; pp. 71:1–71:17



Leibniz International Proceedings in Informatics

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

- *exact* representation:  $f(x) = p(x)$  for all  $x \in \{0, 1\}^n$ ,
- *approximate* representation:  $|f(x) - p(x)| \leq 1/3$  for all  $x \in \{0, 1\}^n$ , and
- *sign* representation:  $(1 - 2f(x))p(x) > 0$  for all  $x \in \{0, 1\}^n$ .

Arguably the most important measure associated with a polynomial is its (total) *degree*. Let  $\deg(f)$ ,  $\widetilde{\deg}(f)$ , and  $\deg_{\pm}(f)$  denote the minimal possible degree of a real polynomial *exactly*, *approximately*, and *sign* representing  $f$ , respectively. These different notions of degrees capture notions of efficiency in many different models of computation (e.g., decision trees, quantum query, perceptrons), and are thus well-studied in literature (see, e.g., [6, 7, 14] and the references therein).

For instance,  $\deg_{\pm}(f)$  (called sign degree) has strong connections to – separations among complexity classes [7], designing efficient learning algorithm [28, 27], and lower bounds against circuits, formulas, communication complexity, etc. [12, 18]. Similarly, upper bounds on  $\widetilde{\deg}(f)$  (called approximate degree), has strong connections to learning theory [25, 29, 37], approximate inclusion-exclusion [24, 43], differentially private data release [47, 17], etc. While the lower bounds on approximate degree lead to lower bounds in quantum query complexity [5, 2, 1], communication complexity [43, 38], circuit complexity [3], etc.

Despite decades of work in this area, there are many important problems that are yet to be resolved completely. One such problem pertains to the composition of approximate degrees. For any two Boolean functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$ , define the *composed* function  $f \circ g: \{0, 1\}^{nm} \rightarrow \{0, 1\}$  as follows

$$f \circ g(x_{11}, \dots, x_{1m}, \dots, x_{n1}, \dots, x_{nm}) = f(g(x_1), \dots, g(x_n)),$$

where  $x_i = (x_{i1}, \dots, x_{im}) \in \{0, 1\}^m$  for  $i \in [n]$ . The function  $f$  is called the outer function and  $g$  the inner function.

Investigating the behaviour of complexity measures under composition has been a quintessential tool in our quest to gain insights into relationships among different measures. In particular, composition has been used successfully on numerous occasions to show separations between various complexity measures associated with Boolean functions, see, e.g., [36, 33, 23, 4, 46, 19]. A big open problem in this context is to understand how approximate degree behaves under composition. More formally, it asks whether for all Boolean functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$ ,

$$\widetilde{\deg}(f \circ g) = \widetilde{\Theta}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))?$$

The tilde in the  $\widetilde{\Theta}$  notation hides a factor polynomial in  $\log(n + m)$ . This problem is often referred to as the “approximate degree composition” problem.

The upper bound,  $\widetilde{\deg}(f \circ g) = O(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))$ , was established in a seminal work [42] of Sherstov. Thus to completely resolve the problem it remains to prove a matching lower bound on the approximate degree of a composed function in terms of the approximate degree of the individual functions. In other words, does the following hold for all Boolean functions  $f$  and  $g$ ,

$$\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(g))?$$

In this article we will refer to the aforementioned (lower bound) question by the phrase “approximate degree composition” problem.

Numerous works, including those by [33, 4, 39, 41, 40, 13, 8, 15], actively pursued these lower bounds, leading to newer connections with several important problems in the field. However, establishing the lower bound  $\widetilde{\deg}(f \circ g) = \widetilde{\Omega}(\widetilde{\deg}(f) \widetilde{\deg}(g))$  even for specific

functions or restricted classes of functions is often very challenging. For example, consider the composed function  $\text{OR} \circ \text{AND}$ , it took a long series of work [33, 44, 4, 41, 40, 13] over nearly two decades to prove that  $\widetilde{\text{deg}}(\text{OR} \circ \text{AND}) = \Omega(\widetilde{\text{deg}}(\text{OR}) \widetilde{\text{deg}}(\text{AND}))$ . Till date we know that the approximate degree composes in the following cases:

- when the outer function  $f$  has full approximate degree, i.e.,  $\Theta(n)$  [39],
- when the outer function  $f$  is a symmetric function [8],
- when the outer function  $f$  has minimal approximate degree with respect to its block sensitivity, i.e.,  $\widetilde{\text{deg}}(f) = O(\sqrt{\text{bs}(f)})$  [15], and
- when the sign degree of the inner function is same as its approximate degree [39, 30].

This work focuses on the behavior of approximate degree when *recursive functions* are composed with other general functions (as outer or inner function). Here, by recursive functions, we mean the functions of the kind  $h^d$  ( $h$  composed with itself  $d$  times) where the arity of  $h$  is small. The function  $h$  is often called the base function and the function  $f$  is called the recursive- $h$  function.

Recursive functions are an important class of Boolean functions that are studied in various different contexts in the analysis of Boolean functions, mainly in proving various lower bounds [4, 45, 36, 33, 34, 9]. For example, the Kushilevitz's function [34] which is the only known non-trivial example of functions with low degree and high sensitivity is a recursive function of a carefully chosen base function. Recursive majority,  $\text{MAJ}_3^d$ , is another recursive function that has been studied extensively in the literature for its different properties [36, 22, 31, 32]. Boppana (see, e.g., [36]) used it to provide the first evidence that the randomized query is more powerful than deterministic query [36]. In the same article, they show a similar separation using recursive  $\text{AND}_2 \circ \text{OR}_2$  function too. In a different application of recursive  $\text{AND}_2 \circ \text{OR}_2$ , [23] show separation between deterministic tree-size complexity and number of monomials in the minimal DNF or CNF.

The approximate degree composition was not known when the outer or inner function is a recursive function, in general. For some special recursive functions, however, it was known that the approximate degree composes. For example, the OR function on  $n = 3^d$  bits is same as  $\text{OR}_3^d$ . After a series of works ([33, 4, 40, 13, 41]), it was proven that the approximate degree composition holds when the outer function is OR, and in general symmetric [8]. Similarly, from the result of [39, 30] it can be observed that the lower bound holds when either the inner or outer function is recursive PARITY. Unfortunately, these results can't be applied in general even when the base function is symmetric or it has full approximate degree.

This scenario leads to the natural question:

*Can we prove that  $\widetilde{\text{deg}}(f \circ g) = \Omega(\widetilde{\text{deg}}(f) \cdot \widetilde{\text{deg}}(g))$  when the outer function  $f$  or the inner function  $g$  is recursive?*

## 1.1 Our Results

Let  $h : \{0, 1\}^k \rightarrow \{0, 1\}$  be a function on  $k$ -bits. Let  $h^d$  denote the Boolean function represented by the complete  $k$ -ary tree of depth  $d$  such that each internal node of the tree is labelled by  $h$  and the leaves of tree are labelled by distinct variables. Our main result shows that the composition theorem holds for any  $h^d$  (except a few specific  $h$ 's), either as the outer function with any inner function or as the inner function with any outer function.

## 71:4 Approximate Degree Composition for Recursive Functions

► **Theorem 1.** *Let  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$  be two Boolean functions and  $d \geq C \log \log n$  for a large enough constant  $C$ . Then,*

$$\widetilde{\deg}(f \circ g) = \Omega \left( \frac{\widetilde{\deg}(f) \widetilde{\deg}(g)}{\text{polylog}(n)} \right),$$

*if either of the following conditions hold:*

1.  $f = h^d$ , for any Boolean function  $h$ .
2.  $g = h^d$ , for any Boolean function  $h$  with constant arity and not equal to AND or OR.

In light of the above theorem, understanding the composition of approximate degree when inner function is OR is the central case for making progress towards the general composition question.

We would like to emphasize that there are not many results which prove composition theorem for a general class of inner functions. Theorem 1 shows that the composition property holds if the inner function is recursive irrespective of the outer function.

We further note that Theorem 1 doesn't follow from the known results even when the composition theorem is known to hold for the base function. Firstly, it is known that the composition lower bound holds when the outer function is symmetric [8]; though, a repeated composition of a symmetric function will incur the factor of  $(\log n)^d$  (because of the  $\log n$  factor hiding in the  $\widetilde{\Omega}$  notation). Secondly, while the majority function,  $\text{MAJ}_n$ , has full approximate degree  $(\Theta(n))$ ,  $\text{MAJ}_3^d$  doesn't have full approximate degree. Thus, Sherstov's result [39] that proves composition theorem holds for functions with full approximate degree cannot be applied in the case of recursive majority. The situation is similar for the inner function as well.

Moving ahead, the proof of Theorem 1 uses two ideas.

- We first prove that a similar theorem works for the specific case of  $h = \text{MAJ}_3$  and  $h = \text{AND}_2 \circ \text{OR}_2$  functions.
- Then, we use a general  $h$  to *simulate*  $\text{AND}_2 \circ \text{OR}_2$ ; hence, proving composition for the general case.

The case of recursive  $h = \text{MAJ}_3$  and  $h = \text{AND}_2 \circ \text{OR}_2$  functions is in itself very interesting. There have been several works towards exploring the approximate degree and other properties of these two functions [21, 26, 36, 23]. Given their importance, and the fact that it is a central step in our main result (Theorem 1), we state the composition theorem for these two functions separately.

► **Theorem 2.** *Let  $f$  and  $g$  be two Boolean functions. Then,*

$$\widetilde{\deg}(f \circ h^d) = \widetilde{\Omega}(\widetilde{\deg}(f) \widetilde{\deg}(h^d)) \text{ and } \widetilde{\deg}(h^d \circ g) = \widetilde{\Omega}(\widetilde{\deg}(h^d) \widetilde{\deg}(g)),$$

*where  $h$  is either  $\text{MAJ}_3: \{0, 1\}^3 \rightarrow \{0, 1\}$  or  $\text{AND}_2 \circ \text{OR}_2: \{0, 1\}^4 \rightarrow \{0, 1\}$ ,  $n$  is the arity of the outer function,  $d \geq C \log \log n$  for a large enough constant  $C$ , and  $\widetilde{\Omega}(\cdot)$  hides  $\text{polylog}(n)$  factors.*

To prove Theorem 2 we will need the following lemma. Even though the lemma can be obtained from a combination of known results (e.g., [39] and [10]) with appropriate parameters, we give a self-contained simpler proof of the lemma, inspired by the primal-dual perspective of [40].

► **Lemma 3.** *For any Boolean functions  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$ ,*

$$\widetilde{\deg}(f \circ \text{MAJ}_t \circ g) = \Omega(\widetilde{\deg}(f) \widetilde{\deg}(g)) \tag{1.1}$$

*for  $t \geq C \log n$  for a large enough constant  $C$ .*

Note that, Lemma 3 gives a way to settle the composition question affirmatively. In particular, if  $\widetilde{\deg}(f \circ \text{MAJ}_t \circ g) = \widetilde{O}(\widetilde{\deg}(f \circ g))$ , where  $t$  is  $\Theta(\log n)$  and  $n$  is the arity of  $f$ , then it follows that the composition holds for  $f$  and  $g$ .

We also highlight that a tighter lower bound can be obtained when the middle function MAJ is replaced by an “amplifier function” in Lemma 3. Define  $H$  to be a *strong hardness amplifier function* for  $g$  if

$$\widetilde{\deg}_{\frac{1-2^{-\Omega(t)}}{2}}(H \circ g) = \Omega(\widetilde{\deg}(H) \circ \widetilde{\deg}(g)).$$

We also observe that,

$$\widetilde{\deg}(f \circ H \circ g) = \Omega(\widetilde{\deg}(f) \widetilde{\deg}(H) \widetilde{\deg}(g)), \quad (1.2)$$

when  $H$  is a strong hardness amplifier function for  $g$ . We discuss this improvement in the full version of the paper [16].

## 1.2 Proof Ideas

To address the lower bound for the composition of two Boolean functions  $f$  and  $g$ ,  $f \circ g$ , we will call  $f$  to be the “outer function” and  $g$  to be the “inner function”. In the case of three layered composed functions ( $f \circ H \circ g$ ), we will call  $H$  to be the “hardness amplifier” and  $f$  and  $g$  to be the outer and inner functions respectively.

**Primal dual approach to composition.** Our proof technique is based on the primal-dual view used by [40] for proving the composition of  $\text{AND}_n \circ \text{OR}_n$ . Here, instead of using “dual-composition method” (see [13, 14]) we will be using only the dual witness of the inner function. The primal-dual approach is to construct an approximating polynomial for  $f$  with smaller degree than  $\widetilde{\deg}(f)$  by applying a linear operator  $L$  on the assumed approximating polynomial for  $f \circ g$  (say  $p$ , with smaller degree than claimed), leading to a contradiction. The linear operator  $L$  is defined by taking the input to  $f$ , extending it to a probability distribution (which depends upon the dual of  $g$ ) over the inputs of  $f \circ g$  and outputting the expectation.

Let  $\psi$  be the dual witness of  $g$ , we get  $\mu_0$  and  $\mu_1$  by restricting  $\psi$  on support which takes positive and negative values respectively; by the properties of dual witness,  $\mu_1$  (and  $\mu_0$ ) will mostly be supported on inputs  $x$  such that  $g(x) = 1$  (and  $g(x) = 0$  respectively). The input to  $f$  is expanded bit by bit using  $\mu_0$  and  $\mu_1$ , creating a distribution on inputs of  $f \circ g$ .

Formally,  $L$  takes a general function  $h : \{0, 1\}^{mn} \rightarrow \{0, 1\}$  and gives  $Lh : \{0, 1\}^n \rightarrow \mathbb{R}$ .

$$Lh(z_1, \dots, z_n) = \mathbb{E}_{x_1 \sim \mu_{z_1}} \mathbb{E}_{x_2 \sim \mu_{z_2}} \cdots \mathbb{E}_{x_n \sim \mu_{z_n}} [h(x_1, x_2, \dots, x_n)], \quad (1.3)$$

where  $x_i \in \{0, 1\}^m$  for all  $i \in \{1, 2, \dots, n\}$ .

To complete the proof, the following two properties of  $L$  are required:

1. Showing that the polynomial  $Lp$  indeed approximates  $f$  in  $l_\infty$  norm. Intuitively this happens because the restricted distributions ( $\mu_0$  and  $\mu_1$ ) are a pretty good indicator of the value of  $g$ .
2. The degree of  $Lp$  is small, intuitively because  $L$  reduces the degree of every monomial by a factor of  $\widetilde{\deg}(g)$ .

**Problem with the primal dual approach.** Unfortunately, the recipe described above doesn't work well in general due to the error introduced by the expectation over  $\mu_0$  and  $\mu_1$  in the string  $(z_1, \dots, z_n)$ . To handle a noisy string in place of a Boolean string, the approximating polynomial  $p$  needs to be robust. A polynomial is robust to noise  $\frac{1}{3}$ , if for all inputs  $x$  and for all  $\Delta \in [-\frac{1}{3}, \frac{1}{3}]^m$ ,  $|p(x) - p(x + \Delta)| < \varepsilon$ .

While any polynomial  $p$  can be made robust up to error  $\varepsilon$  with degree at most  $\deg(p) + \log(\frac{1}{\varepsilon})$  (see Theorem 11 by [42]), such polynomials are not known to be multilinear, making the analysis of expectation difficult. [11] gives a robust multilinear polynomial for any Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ ; though, the polynomial is defined on a perturbation matrix of input  $x$  instead of  $x$  itself. We now discuss how to overcome this problem.

We give the proof ideas of Theorem 1, Theorem 2 and Lemma 3 in the reverse order, the way they are obtained from each other.

**Proof idea of Lemma 3.** We will use  $\text{MAJ}_t$  to get past this difficulty; it helps to reduce the noise in the input of  $f$  to error  $\frac{1}{n}$ . Using the fact that any multilinear polynomial on  $n$  variables is robust up to error  $\frac{1}{n}$ , we have our lower bound for the function  $\widetilde{\deg}(f \circ \text{MAJ}_t \circ g)$  where  $t = \Omega(\log n)$ .

**Proof idea of Theorem 2.** Using previously known constructions ([48, 20]),  $\text{MAJ}_{\log n}$  can be projected to  $\text{MAJ}_3^d$  and  $(\text{AND}_2 \circ \text{OR}_2)^d$ , where  $d \geq C \log \log n$ . We now replace  $\text{MAJ}_{\log n}$  in Lemma 3 with these recursive functions; by using the associativity of the composition of functions and the approximate degree upper bound [42], we finish the proof of the theorem. Note that we only lose a factor of  $\text{polylog}(n)$  in the lower bound since we only need to simulate  $\text{MAJ}_{\log n}$ .

Now we give the idea about how to replace  $\text{AND}_2 \circ \text{OR}_2$  with almost any recursive function to get our main result.

**Proof idea of Theorem 1.** Given Theorem 2, it is natural to ask, what other recursive functions satisfy the composition property. We show that almost any  $h$  can be used to replace the  $\text{AND}_2 \circ \text{OR}_2$  function. This is done by simulating  $\text{AND}_2$  and  $\text{OR}_2$  using restrictions of  $h$  and its powers. The proof of this simulation is divided into two cases: monotone  $h$  and non-monotone  $h$ .

For the monotone case (except when  $h$  is  $\text{AND}$  or  $\text{OR}$ ): We show that both  $\text{AND}_2$  and  $\text{OR}_2$  will be present as sub-cubes of the original Boolean hypercube of  $h$ .

For the non-monotone case (except when  $h$  is  $\text{PARITY}$  or  $\neg\text{PARITY}$ ): The proof requires more work here because of these two issues. First, there need not be both functions  $\text{AND}_2$  and  $\text{OR}_2$  as sub-cubes (though, we show that at least one will be present). Second, the sub-cube could be rotated. The resolution to both these issues is same. We use the non-monotonicity to construct the negation function. This allows us to rotate the sub-cube as well as construct  $\text{AND}_2/\text{OR}_2$  from the other one.

A slight technical point to note is that when  $h$  is a non-constant arity function and  $h^d$  is the inner function, then the loss in the lower bound will be larger than  $\text{polylog}(n)$ . However, even for the case when the base function  $h$  has arity that is a "slowly" growing function of  $n$  we still obtain a non-trivial lower bound composition result.

The remaining cases of Theorem 1, that is,

(i) when  $f$  or  $g$  equals  $h^d$  for  $h \in \{\text{PARITY}, \neg\text{PARITY}\}$  follows from [39], and (ii) when  $f = h^d$  and  $h \in \{\text{AND}, \text{OR}\}$  follows from [8].

## 2 Notations and Preliminaries

In this paper, we will assume a Boolean function has domain  $\{0, 1\}^n$  and range  $\{0, 1\}$ . We start with some of the important definitions.

► **Definition 4** (Generalized Composition of functions). *For any Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and  $n$  Boolean functions  $g_1, g_2, \dots, g_n$ , define the composed function*

$$f \circ (g_1, g_2, \dots, g_n)(x_1, x_2, \dots, x_n) = f(g_1(x_1), g_2(x_2), \dots, g_n(x_n)),$$

where  $g_i$ 's can have different arities and  $x_i \in \text{Dom}(g_i)$  for all  $i \in [n]$ .

When all the copies of  $g_i$  are the same function  $g$  then the composed function is denoted by  $f \circ g$ .

► **Definition 5** (Recursive functions). *For any Boolean function  $f : \{0, 1\}^t \rightarrow \{0, 1\}$  we define recursive function  $f^d : \{0, 1\}^{t^d} \rightarrow \{0, 1\}$  by  $f^d = \underbrace{f \circ f \circ \dots \circ f}_{d \text{ times}}$ .*

► **Definition 6** (Approximate degree ( $\widetilde{\text{deg}}$ )). *For some constant  $0 < \varepsilon < 1/2$ , a polynomial  $p : \mathbb{R}^n \rightarrow \mathbb{R}$  is said to  $\varepsilon$ -approximate a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  if for all  $x \in \{0, 1\}^n$ ,  $|p(x) - f(x)| \leq \varepsilon$ . The  $\varepsilon$ -approximate degree of  $f$ ,  $\widetilde{\text{deg}}_\varepsilon(f)$ , is the minimum possible degree of a polynomial that  $\varepsilon$ -approximates  $f$ . Conventionally we use  $\widetilde{\text{deg}}(\cdot)$  as the shorthand for  $\widetilde{\text{deg}}_{1/3}(\cdot)$ .*

Note that the constant  $\varepsilon$  in the above definition can be replaced by any constant strictly smaller than  $1/2$  which changes  $\widetilde{\text{deg}}_\varepsilon(f)$  by only a constant factor. We note this well-known fact about error reduction.

► **Lemma 7** (Error reduction). *For any  $\varepsilon > 0$ ,  $\widetilde{\text{deg}}_\varepsilon(f) = \Theta_\varepsilon(\widetilde{\text{deg}}(f))$ , where  $\Theta_\varepsilon(\cdot)$  denotes that the constant in  $\Theta(\cdot)$  depends on  $\varepsilon$ .*

► **Lemma 8** ([38, 39]). *Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  be a function and  $\varepsilon > 0$ . Then,  $\widetilde{\text{deg}}_\varepsilon(f) \geq d$  iff there exists a function  $\psi : \{0, 1\}^n \rightarrow \mathbb{R}$  such that*

$$\sum_{x \in \{0, 1\}^n} |\psi(x)| = 1, \tag{2.1}$$

$$\sum_{x \in \{0, 1\}^n} \psi(x) \cdot f(x) > \varepsilon, \text{ and} \tag{2.2}$$

$$\sum_{x \in \{0, 1\}^n} \psi(x) \cdot p(x) = 0 \text{ for every polynomial } p \text{ of degree } < d. \tag{2.3}$$

In a seminal work, Sherstov [42] showed that the approximate degree can increase at most multiplicatively under composition.

► **Theorem 9** ([42]). *For all Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g : \{0, 1\}^m \rightarrow \{0, 1\}$ ,  $\widetilde{\text{deg}}(f \circ g) = O(\widetilde{\text{deg}}(f) \cdot \widetilde{\text{deg}}(g))$ .*

At times we will be working with inputs that are not Boolean *but* are close to Boolean. So we would also need the following notion of *robust* approximating polynomials.

► **Definition 10** ( $(\delta, \varepsilon)$ -robust approximating polynomial). *Let  $p : \{0, 1\}^m \rightarrow [0, 1]$  be a polynomial. Then, for  $\delta, \varepsilon > 0$ , a  $(\delta, \varepsilon)$ -robust approximating polynomial for  $p$  is a polynomial  $p_{\text{robust}} : \mathbb{R}^m \rightarrow \mathbb{R}$  such that for all  $x \in \{0, 1\}^m$  and for all  $\Delta \in [-\delta, \delta]^m$ ,*

$$|p(x) - p_{\text{robust}}(x + \Delta)| < \varepsilon.$$

## 71:8 Approximate Degree Composition for Recursive Functions

Sherstov [42] proved that for any Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  there exists a robust approximating polynomial with degree at most  $O(\widetilde{\deg}(f) + \log(1/\varepsilon))$ .

► **Theorem 11** ([42]). *A  $(\delta, \varepsilon)$ -robust approximating polynomial for  $p : \{0, 1\}^n \rightarrow [0, 1]$  of degree  $O_\delta(\deg(p) + \log(1/\varepsilon))$  exists. Here  $O_\delta(\cdot)$  denotes that the constant in  $O(\cdot)$  depends on  $\delta$ .*

Note that a robust approximating polynomial need not to be multilinear. For our purposes, we need a multilinear robust approximating polynomial.

► **Theorem 12** (Folklore). *Any multilinear polynomial  $p : \{0, 1\}^n \rightarrow \{0, 1\}$  is  $(\frac{\delta}{n}, \delta)$ -robust.*

A proof of the theorem above can be found at [11, Lemma 3].

► **Theorem 13** ([8]). *For any symmetric Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and any Boolean function  $g : \{0, 1\}^m \rightarrow \{0, 1\}$ ,*

$$\widetilde{\deg}(f \circ g) = \Omega\left(\frac{\widetilde{\deg}(f)\widetilde{\deg}(g)}{\log n}\right).$$

Finally, we define projection of functions.

► **Definition 14** (Projection of functions). *Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  and  $g : \{0, 1\}^m \rightarrow \mathbb{R}$  be two functions. We say that  $f$  is a projection of  $g$ , denoted  $f \leq_{\text{proj}} g$ , iff*

$$f(x_1, \dots, x_n) = g(a_1, \dots, a_m)$$

for some  $a_i \in \{0, 1\} \cup \{x_1, x_2, \dots, x_n\}$ . That is,  $f$  is obtained from  $g$  by substitutions of variables of  $g$  by variables of  $f$  or constants in  $\{0, 1\}$ .

We need the following theorems about computing  $\text{MAJ}_n$  using a projection of recursive functions.

► **Theorem 15** ([20]). *There exists a constant  $C > 0$ , such that  $\text{MAJ}_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is a projection of  $\text{MAJ}_3^d$  where  $d = C \log n$ .*

► **Theorem 16** ([48]). *There exists a constant  $C > 0$ , such that  $\text{MAJ}_n : \{0, 1\}^n \rightarrow \{0, 1\}$  is a projection of  $(\text{AND}_2 \circ \text{OR}_2)^d$  where  $d = C \log n$ .*

### 3 Composition theorem for recursive Majority and alternating AND-OR trees

In this section we give a proof of Theorem 2. We begin with a proof highlight of Lemma 3. The missing proofs are in the full version of the paper [16].

#### 3.1 Proof of Lemma 3

► **Lemma 3.** *For any Boolean functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g : \{0, 1\}^m \rightarrow \{0, 1\}$ ,*

$$\widetilde{\deg}(f \circ \text{MAJ}_t \circ g) = \Omega(\widetilde{\deg}(f)\widetilde{\deg}(g)) \tag{1.1}$$

for  $t \geq C \log n$  for a large enough constant  $C$ .



**Proof.** We will present a proof inspired by the primal-dual view of [40]. Fix any constant  $0 < \varepsilon < 1/2$ . Let  $h := f \circ \text{MAJ}_t \circ g$  be the composed function, and  $p_h : \{0, 1\}^{ntm} \rightarrow \mathbb{R}$  be an  $\varepsilon$ -approximating polynomial for  $h$ .

Further, define  $d := \widetilde{\deg}_{\frac{1-\varepsilon}{2}}(g)$ . Then, by Lemma 8, there exists a function  $\psi : \{0, 1\}^m \rightarrow \mathbb{R}$  such that

$$\sum_{x \in \{0, 1\}^m} |\psi(x)| = 1, \quad (3.1)$$

$$\sum_{x \in \{0, 1\}^m} \psi(x) \cdot g(x) > \frac{1-\varepsilon}{2}, \text{ and} \quad (3.2)$$

$$\sum_{x \in \{0, 1\}^m} \psi(x) \cdot p(x) = 0 \text{ for every polynomial } p \text{ of degree } < d. \quad (3.3)$$

Let  $\mu$  be the probability distribution on  $\{0, 1\}^m$  given by  $\mu(x) = |\psi(x)|$  for  $x \in \{0, 1\}^m$ . From (3.3), we have  $\sum_{x \in \{0, 1\}^m} \psi(x) = 0$ . Therefore, the sets  $\{x \mid \psi(x) < 0\}$  and  $\{x \mid \psi(x) > 0\}$  are weighted equally by  $\mu$ . Let  $\mu_0$  and  $\mu_1$  be the probability distributions obtained by conditioning  $\mu$  on the sets  $\{x \mid \psi(x) < 0\}$  and  $\{x \mid \psi(x) > 0\}$  respectively. Hence,

$$\mu = \frac{1}{2}\mu_0 + \frac{1}{2}\mu_1, \quad \text{and} \quad \psi = \frac{1}{2}\mu_1 - \frac{1}{2}\mu_0.$$

We note an important property of the distributions  $\mu_0$  and  $\mu_1$  which shows that the error between  $\text{sign}(\psi(x))$  and  $g(x)$  is low.

► **Lemma 17.**  $\mathbb{E}_{x \sim \mu_1}[g(x)] > 1 - \varepsilon$ .

► **Lemma 18.**  $\mathbb{E}_{x \sim \mu_0}[g(x)] < \varepsilon$ .

Consider the following linear operator  $L$  that maps functions  $h : \{0, 1\}^{ntm} \rightarrow \mathbb{R}$  to functions  $Lh : \{0, 1\}^n \rightarrow \mathbb{R}$ ,

$$Lh(z) = \mathbb{E}_{\substack{x_{11} \sim \mu_{z_1} \\ x_{12} \sim \mu_{z_1}}} \mathbb{E}_{\substack{x_{21} \sim \mu_{z_2} \\ x_{22} \sim \mu_{z_2}}} \cdots \mathbb{E}_{\substack{x_{n1} \sim \mu_{z_n} \\ x_{n2} \sim \mu_{z_n}}} [h(x_{11}, \dots, x_{1t}, x_{21}, \dots, x_{2t}, \dots, x_{n1}, \dots, x_{nt})]. \quad (3.4)$$

Recall  $h = f \circ \text{MAJ}_t \circ g$  and  $p_h$  be  $\varepsilon$ -approximating polynomial for  $h$ . Thus by convexity of  $L$  we have  $\|L(h - p_h)\|_\infty \leq \varepsilon$ . We will now observe some useful properties of the linear operator  $L$ .

► **Lemma 19.**  $\deg(Lp_h) \leq \deg(p_h)/d$ , where  $d = \widetilde{\deg}_{\frac{1-\varepsilon}{2}}(g)$ .

We now show that  $Lp_h$  is in fact an approximating polynomial for  $f$ .

► **Lemma 20.** Fix  $0 < \delta < 1/2$ . Recall  $p_h$  is an  $\varepsilon$ -approximating polynomial for  $h = f \circ \text{MAJ}_t \circ g$ . Let  $t = \Theta(\log n + \log(1/\delta))$  where the constant in  $\Theta(\cdot)$  depends on  $\varepsilon$ . Then,  $Lp_h$  is a  $(\delta + \varepsilon)$ -approximating polynomial for  $f$ . That is,

$$\|f - Lp_h\|_\infty \leq \|f - Lh\|_\infty + \|Lh - Lp_h\|_\infty \leq \delta + \varepsilon.$$

## 71:10 Approximate Degree Composition for Recursive Functions

**Proof.** It suffices to show  $\|f - Lh\|_\infty \leq \delta$ . To this end, consider  $Lh(z)$ .

$$\begin{aligned} Lh(z) &= \mathbb{E}_{\substack{x_{11} \sim \mu_{z_1} \\ x_{12} \sim \mu_{z_1}}} \mathbb{E}_{\substack{x_{21} \sim \mu_{z_2} \\ x_{22} \sim \mu_{z_2}}} \cdots \mathbb{E}_{\substack{x_{n1} \sim \mu_{z_n} \\ x_{n2} \sim \mu_{z_n}}} [f \circ \text{MAJ}_t \circ g(x_{11}, \dots, x_{1t}, \dots, x_{n1}, \dots, x_{nt})] \\ &\quad \vdots \\ &\quad \mathbb{E}_{\substack{x_{1t} \sim \mu_{z_1} \\ x_{2t} \sim \mu_{z_2}}} \mathbb{E}_{\substack{x_{2t} \sim \mu_{z_2} \\ x_{3t} \sim \mu_{z_3}}} \cdots \mathbb{E}_{\substack{x_{nt} \sim \mu_{z_n}}} \\ &= f \left( \text{MAJ}_t \left( \mathbb{E}_{\mu_{z_1}} [g], \dots, \mathbb{E}_{\mu_{z_1}} [g] \right), \dots, \text{MAJ}_t \left( \mathbb{E}_{\mu_{z_n}} [g], \dots, \mathbb{E}_{\mu_{z_n}} [g] \right) \right) \\ &= f(z'_1, z'_2, \dots, z'_n), \end{aligned}$$

where  $\|z - z'\|_\infty \leq \delta/n$  because  $t = \Theta_\varepsilon(\log n + \log(1/\delta))$  and Lemmas 18 and 17.

Therefore, for any  $z \in \{0, 1\}^n$ ,  $|f(z) - Lh(z)| = |f(z) - f(z')| \leq \delta$ , since  $\|z - z'\|_\infty \leq \delta/n$  and Lemma 12.  $\blacktriangleleft$

Since  $Lp_h$  is a  $(\delta + \varepsilon)$ -approximating polynomial for  $f$ , we also have  $\deg(Lp_h) \geq \widetilde{\deg}_{\delta+\varepsilon}(f)$ . We therefore have the following inequalities

$$\widetilde{\deg}_{\delta+\varepsilon}(f) \leq \deg(Lp_h) \leq \frac{\deg(p_h)}{\widetilde{\deg}_{\frac{1-\varepsilon}{2}}(g)}.$$

Rewriting we have

$$\widetilde{\deg}_\varepsilon(f \circ \text{MAJ}_t \circ g) = \deg(p_h) \geq \widetilde{\deg}_{\delta+\varepsilon}(f) \cdot \widetilde{\deg}_{\frac{1-\varepsilon}{2}}(g). \quad (3.5)$$

This completes the proof of Lemma 3.  $\blacktriangleleft$

### 3.2 Proof of Theorem 2

We note an easy to observe fact about approximate degree of projections of functions.

► **Fact 3.6.** Let  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$  be such that  $f \leq_{\text{proj}} g$ , i.e.,  $f$  is a projection of  $g$ . Then, for any  $\varepsilon \in (0, 1/2)$ ,  $\deg_\varepsilon(f) \leq \deg_\varepsilon(g)$ .

Consider the recursive-majority function  $\text{MAJ}_3^d$  given by the complete 3-ary tree of height  $d$  with internal nodes labeled by  $\text{MAJ}_3$  and the leaves are labeled by distinct variables. Fix  $d \geq C \log \log n$  for a large enough constant  $C$ .

First, observe that  $\text{MAJ}_3^d$  is *not* a symmetric function. Secondly, it doesn't have *full* approximate degree ([35]). And finally, its approximate degree is *not* equal to  $\Theta\left(\sqrt{\text{bs}(\text{MAJ}_3^d)}\right)$  (it follows from the fact that  $\text{bs}(\text{MAJ}_3^d)$  is linear with  $\widetilde{\deg}(\text{MAJ}_3^d)$ . See the full version [16] for a proof of  $\widetilde{\deg}(\text{MAJ}_3^d) = 2^d$ ). Thus, none of the previous works [39, 8, 15] imply that approximate degree composes when one of the (inner or outer) functions is recursive-majority  $\text{MAJ}_3^d$ .

**Proof of Theorem 2.** Let  $\text{MAJ}_3^d$  be the recursive-majority function obtained by the complete 3-ary tree of height  $d$  with internal nodes labeled by  $\text{MAJ}_3$  and the leaves are labeled by distinct variables. Let  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  be an arbitrary function and consider the approximate degree of the composed function  $f \circ \text{MAJ}_t \circ \text{MAJ}_3^d$  where  $t = \Theta(\log n)$ .

$$\widetilde{\deg}(f \circ \text{MAJ}_t \circ \text{MAJ}_3^d) \leq \widetilde{\deg}(f \circ \text{MAJ}_3^{C \log t} \circ \text{MAJ}_3^d) = \widetilde{\deg}(f \circ \text{MAJ}_3^d \circ \text{MAJ}_3^{C \log t}) \quad (3.7)$$

$$= O(\widetilde{\deg}(f \circ \text{MAJ}_3^d) \cdot \widetilde{\deg}(\text{MAJ}_3^{C \log t})) \quad (3.8)$$

$$= O(\widetilde{\deg}(f \circ \text{MAJ}_3^d) \cdot \text{poly}(t)). \quad (3.9)$$

The first inequality in (3.7) follows from the fact that  $\text{MAJ}_t$  is a projection of  $\text{MAJ}_3^{C \log t}$  (Theorem 15) and Fact 3.6. Then (3.8) follows from Theorem 9.

On the other hand, from Lemma 3, for  $t = \Omega(\log n)$  we have

$$\widetilde{\deg}(f \circ \text{MAJ}_t \circ \text{MAJ}_3^d) = \Omega(\widetilde{\deg}(f) \cdot \widetilde{\deg}(\text{MAJ}_3^d)).$$

Combining with (3.9), we obtain the lower bound

$$\widetilde{\deg}(f \circ \text{MAJ}_3^d) = \Omega\left(\frac{\widetilde{\deg}(f) \cdot \widetilde{\deg}(\text{MAJ}_3^d)}{\text{polylog}(n)}\right).$$

A similar argument shows the following inequalities, where in the last two inequalities we use Theorem 16 instead of Theorem 15, for  $d = \Omega(\log n)$ ,

- $\widetilde{\deg}(\text{MAJ}_3^d \circ f) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}(\text{MAJ}_3^d))$ ,
- $\widetilde{\deg}(f \circ (\text{AND}_2 \circ \text{OR}_2)^d) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}((\text{AND}_2 \circ \text{OR}_2)^d))$ , and
- $\widetilde{\deg}((\text{AND}_2 \circ \text{OR}_2)^d \circ f) = \widetilde{\Omega}(\widetilde{\deg}(f) \cdot \widetilde{\deg}((\text{AND}_2 \circ \text{OR}_2)^d))$ . ◀

## 4 Composition theorem for recursive functions

In this section we prove our main theorem (Theorem 1). It shows that the approximate degree composes when either the inner function or the outer function is a recursive function. More formally,

► **Theorem 1.** *Let  $f: \{0, 1\}^n \rightarrow \{0, 1\}$  and  $g: \{0, 1\}^m \rightarrow \{0, 1\}$  be two Boolean functions and  $d \geq C \log n$  for a large enough constant  $C$ . Then,*

$$\widetilde{\deg}(f \circ g) = \Omega\left(\frac{\widetilde{\deg}(f) \widetilde{\deg}(g)}{\text{polylog}(n)}\right),$$

if either of the following conditions hold:

1.  $f = h^d$ , for any Boolean function  $h$ .
2.  $g = h^d$ , for any Boolean function  $h$  with constant arity and not equal to AND or OR.

The following cases of Theorem 1 follows from prior works:

1.  $f$  or  $g$  equals  $h^d$  for  $h \in \{\text{PARITY}, \neg\text{PARITY}\}$  [39].
2.  $f = h^d$  and  $h \in \{\text{AND}, \text{OR}\}$  [8].

Therefore, it remains to prove Theorem 1 when  $h \notin \{\text{PARITY}, \neg\text{PARITY}, \text{AND}, \text{OR}\}$ . A crucial technical insight that makes the proof work is that when  $h \notin \{\text{PARITY}, \neg\text{PARITY}, \text{AND}, \text{OR}\}$  then  $\text{AND}_2$  and  $\text{OR}_2$  are projections of  $h^3$ . We can thus simulate MAJ using a small power of  $h$ . Thereafter, Lemma 3 is used to conclude Theorem 1. We now work out the details. We first state the main technical lemma we need for Theorem 1 and then complete the proof of the theorem. Finally, we prove the technical lemma in Section 4.1.

► **Lemma 21.** *Let  $h: \{0, 1\}^t \rightarrow \{0, 1\}$  (where  $t \geq 2$ ) be a Boolean function which depends on all  $t$  variables and is not equal to PARITY/ $\neg$ PARITY/OR/AND. The function  $\text{AND}_2$  (and similarly  $\text{OR}_2$ ) can be obtained by setting all but two variables to constants in  $h^k$  for  $k \leq 3$ .*

We now present the proof of Theorem 1 using Lemma 21.

**Proof of Theorem 1.** Let  $h: \{0, 1\}^t \rightarrow \{0, 1\}$  be any Boolean function such that  $h \notin \{\text{PARITY}, \neg\text{PARITY}, \text{AND}, \text{OR}\}$ . We know from Lemma 3 that  $\widetilde{\deg}(f \circ \text{MAJ}_k \circ h^d) = \Omega(\widetilde{\deg}(f) \widetilde{\deg}(h^d))$  where  $k = \Theta(\log n)$ . Like in the proof of Theorem 2, we will simulate  $\text{MAJ}_k$  using  $h^\ell$  for sufficiently large  $\ell$ . From Lemma 21, it follows that  $(\text{AND}_2 \circ \text{OR}_2)^\ell$  is a projection of  $h^{6\ell}$ . Therefore, we obtain from Theorem 16 that  $\text{MAJ}_k$  is a projection of  $h^{C \log k}$  for some constant  $C > 0$ . We thus have the following sequence of inequalities,

$$\begin{aligned}
 \widetilde{\deg}(f \circ h^d) &\geq \widetilde{\deg}(f \circ \text{MAJ}_k \circ h^{(d-C \log k)}) \\
 &= \Omega(\widetilde{\deg}(f) \widetilde{\deg}(h^{(d-C \log k)})) \\
 &= \Omega\left(\frac{\widetilde{\deg}(f) \widetilde{\deg}(h^d)}{t^{C \log k}}\right) \\
 &= \Omega\left(\frac{\widetilde{\deg}(f) \widetilde{\deg}(h^d)}{\text{polylog}(n)}\right).
 \end{aligned}$$

Note that the last equality above uses the fact that  $t$  is a constant. When  $h^d$  is the outer function then we don't need  $t$  to be a constant, while the rest of the argument remains the same to give

$$\widetilde{\deg}(h^d \circ g) = \Omega\left(\frac{\widetilde{\deg}(h^d) \widetilde{\deg}(g)}{\text{polylog}(n)}\right). \quad \blacktriangleleft$$

This completes the proof of the main theorem. We now present a proof of Lemma 21.

#### 4.1 Proof of the main technical lemma (Lemma 21)

We proceed by proving an intermediate result (Lemma 22) before going to the proof of Lemma 21.

Suppose we are allowed to *modify* a Boolean function by two operations: negating some of its variables, and restricting some of the variables to constant values. Lemma 22 proves that almost every Boolean function can be modified to either an AND<sub>2</sub> or an OR<sub>2</sub> function. A restriction of the variables amounts to looking at a smaller hypercube translated to a new point, and negating a variable amounts to rotating the smaller hypercube. In other words, we want to show that there is a *shifted* AND<sub>2</sub> or OR<sub>2</sub> in the Boolean hypercube of  $h$  (see Figure 1 for an example).

This shifted AND<sub>2</sub>/OR<sub>2</sub> in the Boolean hypercube of a Boolean function can be concretely defined by the concept of a sensitive block. For a block of variables  $S \subseteq [n]$  and an input  $x \in \{0, 1\}^n$ , define  $x^{\oplus S} \in \{0, 1\}^n$  to be the input which flips exactly the variables in  $S$  at the input  $x$ . Given a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , a block  $S$  is called sensitive on  $x$  iff  $f(x) \neq f(x^{\oplus S})$ . A block  $S$  is called *minimal sensitive* for  $x$  at  $f$ , if no subset of  $S$  is sensitive for  $x$  at  $f$ .

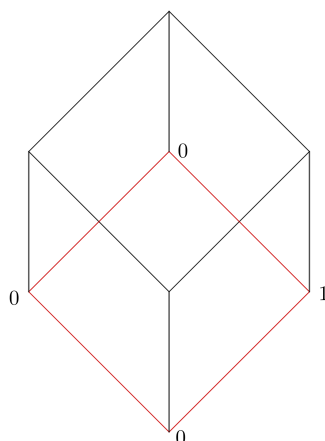
Notice that a shifted AND<sub>2</sub>/OR<sub>2</sub> is a square with three vertices labelled 0 and one vertex labelled 1 or vice versa. This gives us a minimal sensitive block on the vertex opposite to the unique value. It can be easily verified that the converse is also true. So, we define a function to have a shifted AND<sub>2</sub>/OR<sub>2</sub> iff it has a minimal sensitive block of size 2.

We show below that almost all functions have a minimal sensitive block of size 2.

► **Lemma 22.** *Let  $h : \{0, 1\}^t \rightarrow \{0, 1\}$  (where  $t \geq 2$ ) be a Boolean function which depends on all  $t$  variables and is not equal to PARITY/ $\neg$ PARITY. Then, there exists an  $x \in \{0, 1\}^t$  such that  $h$  has a minimal sensitive block of size 2 on  $x$ .*

**Proof.** We will prove the result using induction on the variables. The statement can be easily verified for  $t = 2$ .

Define  $g_0$  (and  $g_1$ ) to be the restrictions of  $h$  by setting  $x_t = 0$  (and  $x_t = 1$ ) respectively. Let  $e_y$  be the edge  $((y, 0), (y, 1))$  in the Boolean hypercube, and  $S_t := \{e_y : y \in \{0, 1\}^{t-1}\}$ . Color an edge  $e_y$  red if  $g_0(y) = g_1(y)$ , and blue otherwise.



■ **Figure 1** A function on 3 bits with a shifted OR marked with red edges.

Notice that not all the edges in  $S_t$  can be red, otherwise  $h$  does not depend on  $x_t$ . Suppose all the edges in  $S_t$  are blue, i.e.  $g_1 = \neg g_0$  (in other words,  $h = g_0 \oplus x_t$ ). Since  $h$  depends on all variables, then  $g_0$  depends on all variables  $x_1, x_2, \dots, x_{t-1}$ . If  $g_0$  is PARITY/ $\neg$ PARITY, then  $h$  is also PARITY/ $\neg$ PARITY. Implying that  $g_0$  is dependent on all its variables and is not PARITY/ $\neg$ PARITY. By induction, there exists a minimal sensitive block of size 2 for  $g_0$  (and hence  $h$ ).

For the rest of the proof, we can assume that there exists both a red and a blue edge in  $S_t$ .

Let  $e_x$  be red and  $e_y$  be blue, this means that  $g_0(x) = g_1(x)$  but  $g_0(y) \neq g_1(y)$ . If  $x$  and  $y$  were at Hamming distance 1, then vertices  $(x, 0), (x, 1), (y, 0)$  and  $(y, 1)$  will give us the required minimal sensitive block of size 2.

If  $x, y$  are not at Hamming distance 1, look at any path from  $x$  to  $y$  in the  $t-1$  dimensional hypercube, say  $z_0 = x, z_1, z_2, \dots, z_l = y$ . The edge  $e_{z_0}$  is red and  $e_{z_l}$  is blue. Since the color needs to switch at some point, there exist  $z_i, z_{i+1}$  at Hamming distance 1 such that  $e_{z_i}$  is red and  $e_{z_{i+1}}$  is blue. Again, the vertices  $(z_i, 0), (z_i, 1), (z_{i+1}, 0)$  and  $(z_{i+1}, 1)$  will give us the required minimal sensitive block of size 2. ◀

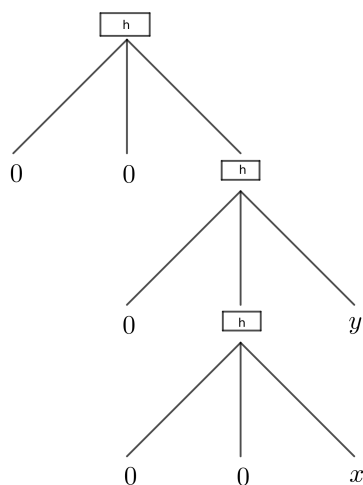
We are prepared to prove Lemma 21 which shows: given a Boolean function  $h$ ,  $\text{AND}_2$  (and  $\text{OR}_2$ ) can be obtained by restricting some of the variables to constants in a very small power of  $h$ . Compared to Lemma 22, we need to remove negation and simulate both  $\text{AND}_2$  and  $\text{OR}_2$  and not just one of them.

We just show how to obtain  $\text{AND}_2$ , the case for  $\text{OR}_2$  is similar. We handle the case of  $h$  being monotone and non-monotone separately.

### Monotone $h$

This case is simpler, and  $\text{AND}_2$  can be obtained as a restriction of  $h$  itself. Let a minimal 1-input be a  $x \in \{0, 1\}^t$  such that setting any 1 bit of  $x$  to 0 changes the value of  $h$ . If there is a minimal 1-input  $x$  of Hamming weight more than 2, we get a  $\text{AND}_2$  by choosing any two indices which are 1 in  $x$ . The following claim finishes the proof for monotone functions.

▷ **Claim 23.** Let  $h : \{0, 1\}^t \rightarrow \{0, 1\}$  be a monotone Boolean function which depends on all variables. If there is no minimal 1-input with Hamming weight more than 2, then  $h$  is the OR function.



■ **Figure 2** An example for constructing  $\text{AND}_2$  using a non-monotone function. Let  $h : \{0,1\}^3 \rightarrow \{0,1\}$  be 0 at  $x = 001$  and 1 otherwise. Use the shifted  $\text{OR}_2$ /minimal sensitive block at 001 with indices  $\{2,3\}$ .

Proof. By abusing the notation, let 0 denote the all 0 input. Since the function is monotone but not constant, we know that  $h(0) = 0$ . Let  $S \subseteq [t]$  capture the indices such that the corresponding Hamming weight 1-input has function value 0,

$$S = \{i : h(0^{\oplus i}) = 0\}.$$

For a  $y \in \{0,1\}^t$ , if the set of 1-indices are not a subset of  $S$ , then  $h(y) = 1$  by monotonicity. If the set of 1-indices are a subset of  $S$ , then  $h(y) = 0$  because there is no minimal 1-input with Hamming weight more than 2.

In other words,  $h$  is the OR function on the remaining  $[t] \setminus S$  variables. Since  $h$  depends on all the  $t$  variables,  $h$  is the OR function.  $\triangleleft$

### Non-monotone $h$

Since  $h$  is a non-monotone function, there exists an input  $a \in \{0,1\}^t$  and an index  $i \in [t]$  such that  $h(a) = 1$ ,  $a_i = 0$  and  $h(a^{\oplus i}) = 0$ . Restricting the variables according to  $a$  (except the  $i$ -th bit) gives  $h_1(x_i) = \neg x_i$ .

From Lemma 22, there exists a  $b \in \{0,1\}^t$  such that  $h$  has a minimal sensitive block of size 2 on  $b$  (shifted  $\text{AND}_2/\text{OR}_2$ ). The main idea of this proof is to use negation and this shifted  $\text{AND}_2/\text{OR}_2$  (Figure 2 gives an example).

For the formal proof, without loss of generality assume that the block have indices 1, 2 (that means  $h(b) = h(b^{\oplus\{1\}}) = h(b^{\oplus\{2\}}) \neq h(b^{\oplus\{1,2\}})$ ). We will finish the proof by considering the two cases  $h(b) = 0$  and  $h(b) = 1$ .

- $h(b) = 0$  (shifted  $\text{AND}_2$ ): Suppose  $b_1 = 0$  and  $b_2 = 1$  (other cases can be handled similarly). Notice that  $\text{AND}_2(x, y) = h(x, \neg y, b_3, \dots, b_t)$ , giving us  $\text{AND}_2(x, y) = h(x, h_1(y), b_3, \dots, b_t)$ .
- $h(b) = 1$  (shifted  $\text{OR}_2$ ): Suppose  $b_1 = 1$  and  $b_2 = 0$  (other cases can be handled similarly). Notice that  $\text{OR}_2(x, y) = h(x, \neg y, b_3, \dots, b_t)$ ; using De Morgan's law,

$$\text{AND}_2(x, y) = \neg \text{OR}_2(\neg x, \neg y) = \neg h(\neg x, y, b_3, \dots, b_t) = h_1(h(h_1(x), y, b_3, \dots, b_t))$$

Since  $h_1$  is also a restriction of  $h$ , the proof is complete.

## 5 Conclusion

Towards the main open problem of approximate degree composition, we have the following immediate question in light of Lemma 3. Can we upper bound  $\text{deg}(f \circ \text{MAJ}_t \circ g)$  in terms of  $\widetilde{\text{deg}}(f \circ g)$ ? Precisely,

► **Open question 24.** *Is  $\widetilde{\text{deg}}(f \circ \text{MAJ}_t \circ g) = \widetilde{O}(\widetilde{\text{deg}}(f \circ g))$ , where  $t = \Theta(\log n)$  and  $n$  is the arity of the outer function  $f$ ?*

Observe that an affirmative solution to the above question solves the composition question for approximate degree in positive. Another interesting question is to find other classes of functions for which the analogue of Equation 1.2 holds.

► **Open question 25.** *Find non-trivial classes of functions  $H$  such that  $\widetilde{\text{deg}}(f \circ h \circ g) = \widetilde{\Omega}(\text{deg}(f) \cdot \text{deg}(h) \cdot \text{deg}(g))$  for all  $h \in H$ ?*

It has the following two useful implications. First, this gives composition for functions  $h \in H$ . In particular, when one of the functions  $h$  (inner or outer) belongs to the class  $H$  then  $\widetilde{\text{deg}}(f \circ h \circ g) = \widetilde{\Omega}(\text{deg}(f) \cdot \text{deg}(h) \cdot \text{deg}(g))$  along with Theorem 9 implies

$$\widetilde{\text{deg}}(h \circ g) = \widetilde{\Omega}(\widetilde{\text{deg}}(h) \cdot \widetilde{\text{deg}}(g)) \quad \text{and} \quad \widetilde{\text{deg}}(f \circ h) = \widetilde{\Omega}(\widetilde{\text{deg}}(f) \cdot \widetilde{\text{deg}}(h)).$$

Second, a function  $h \in H$  can be used as “hardness amplifier” functions.

Another very interesting question that may provide us insights to make progress towards the main question of approximate degree composition is to prove that approximate degree composes when the inner function is  $\text{OR}$ .

► **Open question 26.** *Show that  $\widetilde{\text{deg}}(f \circ \text{OR}) = \widetilde{\Omega}(\widetilde{\text{deg}}(f) \cdot \widetilde{\text{deg}}(\text{OR}))$ .*

---

## References

- 1 S. Aaronson. Impossibility of Succinct Quantum Proofs for Collision-Freeness. *Quantum Inf. Comput.*, 12(1-2):21–28, 2012. doi:10.26421/QIC12.1-2-3.
- 2 S. Aaronson and Y. Shi. Quantum Lower Bounds for the Collision and the Element Distinctness Problems. *J. ACM*, 51(4):595–605, 2004. doi:10.1145/1008731.1008735.
- 3 E. Allender. A Note on the Power of Threshold Circuits. In *FOCS*, pages 580–584, 1989. doi:10.1109/SFCS.1989.63538.
- 4 A. Ambainis. Polynomial Degree and Lower Bounds in Quantum Complexity: Collision and Element Distinctness with Small Range. *Theory Comput.*, 1(1):37–46, 2005. doi:10.4086/toc.2005.v001a003.
- 5 R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum Lower Bounds by Polynomials. *J. ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.

- 6 R. Beigel. The Polynomial Method in Circuit Complexity. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 82–95, 1993.
- 7 R. Beigel. Perceptrons, PP, and the Polynomial Hierarchy. *Comput. Complex.*, 4:339–349, 1994. doi:10.1007/BF01263422.
- 8 S. Ben-David, A. Bouland, A. Garg, and R. Kothari. Classical Lower Bounds from Quantum Upper Bounds. In *FOCS*, pages 339–349, 2018. doi:10.1109/FOCS.2018.00040.
- 9 S. Ben-David, P. Hatami, and A. Tal. Low-Sensitivity Functions from Unambiguous Certificates. In *ITCS*, volume 67, pages 28:1–28:23, 2017. doi:10.4230/LIPIcs.ITCS.2017.28.
- 10 A. Bouland, L. Chen, D. Holden, J. Thaler, and P. N. Vasudevan. On the Power of Statistical Zero Knowledge. In *FOCS*, pages 708–719, 2017. doi:10.1109/FOCS.2017.71.
- 11 H. Buhrman, I. Newman, H. Röhrig, and R. de Wolf. Robust Polynomials and Quantum Algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007. doi:10.1007/S00224-006-1313-Z.
- 12 H. Buhrman, N. K. Vereshchagin, and R. de Wolf. On Computation and Communication with Small Bias. In *CCC*, pages 24–32, 2007. doi:10.1109/CCC.2007.18.
- 13 M. Bun and J. Thaler. Dual Lower Bounds for Approximate Degree and Markov-Bernstein Inequalities. In *ICALP*, volume 7965, pages 303–314, 2013. doi:10.1007/978-3-642-39206-1\_26.
- 14 M. Bun and J. Thaler. Approximate Degree in Classical and Quantum Computing. *Found. Trends Theor. Comput. Sci.*, 15(3-4):229–423, 2022. doi:10.1561/0400000107.
- 15 S. Chakraborty, C. Kayal, R. Mittal, M. Paraashar, S. Sanyal, and N. Saurabh. On the Composition of Randomized Query Complexity and Approximate Degree. In *APPROX/RANDOM*, volume 275, pages 63:1–63:23, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.63.
- 16 Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, and Nitin Saurabh. Approximate Degree Composition for Recursive Functions, 2024. arXiv:2407.08385.
- 17 K. Chandrasekaran, J. Thaler, J. R. Ullman, and A. Wan. Faster Private Release of Marginals on Small Databases. In *ITCS*, pages 387–402, 2014. doi:10.1145/2554797.2554833.
- 18 L. Chen. Adaptivity vs. Postselection, and Hardness Amplification for Polynomial Approximation. In *ISAAC*, volume 64 of *LIPICS*, pages 26:1–26:12, 2016. doi:10.4230/LIPICS.ISAAC.2016.26.
- 19 J. Gilmer, M. Saks, and S. Srinivasan. Composition Limits and Separating Examples for Some Boolean Function Complexity Measures. *Combinatorica*, 36(3):265–311, 2016. doi:10.1007/s00493-014-3189-x.
- 20 O. Goldreich. On (Valiant’s) Polynomial-Size Monotone Formula for Majority. In *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*, pages 17–23. Springer, 2020. doi:10.1007/978-3-030-43662-9\_3.
- 21 M. Göös and T. S. Jayram. A Composition Theorem for Conical Juntas. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPICS*, pages 5:1–5:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.CCC.2016.5.
- 22 T. S. Jayram, R. Kumar, and D. Sivakumar. Two Applications of Information Complexity. In *STOC*, pages 673–682, 2003. doi:10.1145/780542.780640.
- 23 S. Jukna, A. Razborov, P. Savický, and I. Wegener. On P versus  $NP \cap co-NP$  for Decision trees and Read-Once Branching Programs. *Comput. Complex.*, 8(4):357–370, 1999.
- 24 J. Kahn, N. Linial, and A. Samorodnitsky. Inclusion-Exclusion: Exact and Approximate. *Comb.*, 16(4):465–477, 1996. doi:10.1007/BF01271266.
- 25 A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically Learning Halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. doi:10.1137/060649057.
- 26 P. Kamath and P. Vasudevan. Approximate Degree of AND-OR trees, 2014. URL: <https://www.scottaaronson.com/showcase3/kamath-pritish-vasudevan-prashant.pdf>.
- 27 A. Klivans, R. O’Donnell, and R. Servedio. Learning Intersections and Thresholds of Halfspaces. *J. Comput. Syst. Sci.*, 68(4):808–840, 2004. doi:10.1016/J.JCSS.2003.11.002.



- 28 A. Klivans and R. 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.
- 29 A. Klivans and R. Servedio. Toward Attribute Efficient Learning of Decision Lists and Parities. *J. Mach. Learn. Res.*, 7:587–602, 2006. URL: <http://jmlr.org/papers/v7/klivans06a.html>.
- 30 T. Lee. A Note on the Sign Degree of Formulas. *CoRR*, abs/0909.4607, 2009. arXiv:0909.4607.
- 31 N. Leonardos. An Improved Lower Bound for the Randomized Decision Tree Complexity of Recursive Majority. In *ICALP*, volume 7965, pages 696–708, 2013. doi:10.1007/978-3-642-39206-1\_59.
- 32 F. Magniez, A. Nayak, M. Santha, J. Sherman, G. Tardos, and D. Xiao. Improved Bounds for the Randomized Decision tree Complexity of Recursive Majority. *Random Struct. Algorithms*, 48(3):612–638, 2016. doi:10.1002/rsa.20598.
- 33 N. Nisan and M. Szegedy. On the Degree of Boolean Functions as Real Polynomials. *Comput. Complex.*, 4:301–313, 1994. doi:10.1007/BF01263419.
- 34 N. Nisan and A. Wigderson. On Rank vs. Communication Complexity. *Combinatorica*, 15(4):557–565, 1995. doi:10.1007/BF01192527.
- 35 B. Reichardt and R. Špalek. Span-Program-Based Quantum Algorithm for Evaluating Formulas. *Theory Comput.*, 8(1):291–319, 2012. doi:10.4086/TQC.2012.V008A013.
- 36 M. Saks and A. Wigderson. Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees. In *FOCS*, pages 29–38, 1986. doi:10.1109/SFCS.1986.44.
- 37 R. Servedio, L.-Y. Tan, and J. Thaler. Attribute-Efficient Learning and Weight-Degree Tradeoffs for Polynomial Threshold Functions. In *COLT*, volume 23, pages 14.1–14.19. JMLR.org, 2012. URL: <http://proceedings.mlr.press/v23/servedio12/servedio12.pdf>.
- 38 A. Sherstov. The Pattern Matrix Method. *SIAM J. Comput.*, 40(6):1969–2000, 2011. doi:10.1137/080733644.
- 39 A. Sherstov. Strong Direct Product Theorems for Quantum Communication and Query Complexity. *SIAM J. Comput.*, 41(5):1122–1165, 2012. doi:10.1137/110842661.
- 40 A. Sherstov. Approximating the AND-OR Tree. *Theory Comput.*, 9:653–663, 2013. doi:10.4086/toc.2013.v009a020.
- 41 A. Sherstov. The Intersection of Two Halfspaces has High Threshold Degree. *SIAM Journal on Computing*, 42(6):2329–2374, 2013. doi:10.1137/100785260.
- 42 A. Sherstov. Making Polynomials Robust to Noise. *Theory Comput.*, 9:593–615, 2013. doi:10.4086/TQC.2013.V009A018.
- 43 A. A. Sherstov. Approximate Inclusion-Exclusion for Arbitrary Symmetric Functions. *Comput. Complex.*, 18(2):219–247, 2009. doi:10.1007/S00037-009-0274-4.
- 44 Y. Shi. Approximating Linear Restrictions of Boolean functions, 2002.
- 45 M. Snir. Lower Bounds on Probabilistic Linear Decision Trees. *Theoretical Computer Science*, 38:69–82, 1985. doi:10.1016/0304-3975(85)90210-5.
- 46 A. Tal. Properties and Applications of Boolean Function Composition. In *ITCS*, pages 441–454, 2013. doi:10.1145/2422436.2422485.
- 47 J. Thaler, J. R. Ullman, and S. P. Vadhan. Faster Algorithms for Privately Releasing Marginals. In *ICALP*, volume 7391 of *Lecture Notes in Computer Science*, pages 810–821. Springer, 2012. doi:10.1007/978-3-642-31594-7\_68.
- 48 L. Valiant. Short Monotone Formulae for the Majority Function. *Journal of Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.



# Public Coin Interactive Proofs for Label-Invariant Distribution Properties

Tal Herman  

Weizmann Institute of Science, Rehovot, Israel

---

## Abstract

Assume we are given sample access to an unknown distribution  $D$  over a large domain  $[N]$ . An emerging line of work has demonstrated that many basic quantities relating to the distribution, such as its distance from uniform and its Shannon entropy, despite being hard to approximate through the samples only, can be *efficiently and verifiably* approximated through interaction with an untrusted powerful prover, that *knows* the entire distribution [Herman and Rothblum, STOC 2022, FOCS 2023]. Concretely, these works provide an efficient proof system for approximation of any label-invariant distribution quantity (i.e. any function over the distribution that's invariant to a re-labeling of the domain  $[N]$ ).

In our main result, we present the first efficient *public coin* AM protocol, for any label-invariant property. Our protocol achieves sample complexity and communication complexity of magnitude  $\tilde{O}(N^{2/3})$ , while the proof can be generated in quasi-linear  $\tilde{O}(N)$  time.

On top of that, we also give a public-coin protocol for efficiently verifying the distance a between a samplable distribution  $D$ , and some explicitly given distribution  $Q$ .

**2012 ACM Subject Classification** Theory of computation → Interactive proof systems

**Keywords and phrases** Interactive Proof Systems, Distribution Testing, Public-Coin Protocols

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.72

**Category** RANDOM

**Funding** *Tal Herman*: 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. 819702), from the Israel Science Foundation (grant number 5219/17), and from the Simons Foundation Collaboration on the Theory of Algorithmic Fairness.

## 1 Introduction

Given sample access to a distribution, what can we learn about the distribution, and what is the complexity of learning? These questions are central to computer science and statistics and have guided a rich body of work with applications ranging many fields. An emerging line of work asks the following question:

What is the complexity of *verifying* claims about a samplable distribution?

That is, suppose there exists a powerful yet untrusted prover that claims to have drawn many samples from a distribution  $D$ , and concluded that it satisfies some condition, e.g. its support is of size at most  $K$ , its Shannon entropy is  $h$ , etc. Can a verifier interacting with the prover be convinced that the claim is (approximately) correct, while taking fewer samples and running in less time than required to compute these measures directly from samples?

This question was raised by Chiesa and Gur [5], and recently Herman and Rothblum [14] showed that a rich family of distribution properties, namely *label-invariant* distribution properties - those distribution measures that remain unchanged after permuting the domain (such as the distribution's support size and Shannon entropy) - have (*doubly*) efficient proof systems, that for natural problems, allow verification that is significantly faster than computation from samples only. These protocols are *private-coin* protocols, in which the



© Tal Herman;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 72; pp. 72:1–72:23



Leibniz International Proceedings in Informatics

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

verifier can draw samples from  $D$ , toss random coins, and choose whether to send them to the prover, or keep them hidden from it. Indeed, the protocols in [14] rely heavily on the fact that the verifier *hides* its random coin-tosses in order to perform the verification. In this work we explore *public-coin* protocols for verifying distribution properties, in which the verifier reveals to the prover every coin it tosses immediately upon drawing it. We construct efficient *public-coin* proof systems for label-invariant distribution properties, and more.

More concretely, we follow the definition of *public-coin* proof systems for distribution properties from Chiesa and Gur [5], in which the verifier can only send random coin tosses to the prover, and the samples they draw from  $D$  are independent from the transcript of the protocol, and are drawn only after the communication phase.

Our work studies the power of public-coin proof systems in the context of verifying properties of an unknown samplable distribution. We find this to be a foundational question: indeed, the power of public-coin proof systems has been a central question since they were first introduced [10, 2]. In the classical setting (verifying the membership of a fixed and known input in a language), Goldwasser and Sipser [11] showed how to convert general protocols into public-coin ones (albeit their transformation does not preserve the honest prover’s running time [19, 1]). In our context, where the verifier only has sampling access to the unknown distribution, no such general transformation is known. Chiesa and Gur showed upper and lower bounds for public-coin interactive proofs for distribution properties. Beyond the foundational importance of public-coin protocols, they are also important for removing interaction using the Fiat-Shamir paradigm [7] and for transforming general protocol into zero-knowledge ones [9, 4]

### 1.1 This Work: Public-coin Protocols for Label-Invariant Distribution Properties

Our main result is a new *public-coin* protocol for label-invariant distribution properties. We proceed to present this result, and put it into context with the private-coin setting of [14], and the other public-coin distribution verification protocols of [5].

A distribution property  $\mathcal{P} = (\mathcal{P}_N)_{N \in \mathbb{N}}$  is an ensemble such that  $\mathcal{P}_N$  is a set of distributions over domain  $[N]$ . We consider the distance of a distribution  $D$  over domain  $[N]$  from the property by the *total variation* of  $D$  from the closest distribution to it in  $\mathcal{P}_N$ . A distribution property is said to be *label-invariant* if permuting the domain doesn’t change  $\mathcal{P}$ . This family of distribution properties contains many natural properties, such as the property of being close to uniform over some subset of the domain, or having Shannon entropy roughly  $k$ .

► **Theorem 1** (Main result: public-coin IPs for label-invariant properties, informal). *For every label-invariant distribution property  $\mathcal{P}$  with a doubly-efficient approximate decision procedure,<sup>1</sup> there exists a 2-message public-coin interactive protocol as follows. The prover and the verifier both get as input an integer  $N$  and proximity parameters  $\varepsilon_c, \varepsilon_f \in [0, 1]$  where  $\varepsilon_c < \varepsilon_f$ , as well as sampling access to an unknown distribution  $D$  over support  $[N]$ , and the following properties hold:*

---

<sup>1</sup> See Definition 28. In a nutshell, these are label-invariant properties that can be efficiently decided from the  $\tau$ -approximate bucket-histogram of the distribution, i.e. by only knowing how many elements have probability roughly  $\frac{(1+\tau)^j}{N}$  for all  $j$ , see Definition 5. [13] showed that this assumption is quite mild, and many natural distribution properties admit such a procedure, the reader is referred to [13] for a deeper exploration of this notion.

- *Completeness:* if  $D$  is  $\varepsilon_c$ -close to the property (its total variation distance from the closest distribution in the property is at most  $\varepsilon_c$ ), and the prover follows the protocol, then w.h.p. the verifier accepts.
- *Soundness:* if  $D$  is  $\varepsilon_f$ -far from the property (its total variation distance from every distribution in the property is at least  $\varepsilon_f$ ), then w.h.p. no matter how the prover cheats, the verifier rejects.
- *Doubly-efficient prover:* Taking  $\rho = \varepsilon_f - \varepsilon_c$ , the honest prover's runtime and sample complexity are  $\tilde{O}(N) \cdot \text{poly}(1/\rho)$ .
- *Efficient verification:* the communication complexity and the verifier's sample complexity and runtime are all  $\tilde{O}(N^{2/3}) \cdot \text{poly}(1/\rho)$ .

### Public-coin verification vs. testing of label-invariant distribution properties

Observe that the protocol above allows us to efficiently approximate the distance of  $D$  from  $\mathcal{P}$ , by running a binary search with different values for  $\varepsilon_c, \varepsilon_f$ . Raskhodnikova et al. [17], and Valiant and Valiant [20] showed that approximating the distance between  $D$  and natural label-invariant distribution properties, given only black-box sample access to the distribution, requires  $\Theta(N/\log N)$  samples. This includes approximating the distance from being uniform over the entire domain, from having entropy  $k$ , and more. Thus, our result demonstrates that public-coin verification can be more efficient than stand-alone computation with no access to a prover for these natural distribution problems.

### Comparison with the secret-coin setting of Herman and Rothblum [14]

Herman and Rothblum provided a *secret-coin* interactive proof for verifying membership in any label-invariant distribution property (that admits an efficient approximate decision procedure) with verifier sample complexity, runtime, communication complexity of magnitude  $\tilde{O}(\sqrt{N})$ , and only two messages. The first message in their protocol contains a tuple of elements in  $[N]$ , where each element was sampled with probability  $\frac{1}{2}$  from the distribution  $D$ , and with probability  $\frac{1}{2}$  was drawn uniformly from  $[N]$ . Crucially for their argument, the verifier doesn't share with the prover which samples were drawn according to which distribution, and later capitalizes on that fact to reject dishonest prover behavior.

In our public-coin protocol not only is the verifier required to share the random coin tosses, it also cannot send samples from  $D$  as part of the communication. Thus, Theorem 4 achieves a similar result qualitatively to theirs, but using only public coins, at the cost of more samples and communication.

### Comparison with Chiesa and Gur [5]

Chiesa and Gur provided *public-coin* protocols for any property with communication  $c = \tilde{O}(N)$ , and verifier sample complexity  $s = O(\sqrt{N})$ , by having the prover send an explicit description of the distribution, and the verifier use an *identity tester* from the distribution testing literature to check that the description matches the samplable distribution. Then, the verifier accepts if  $D$  is both close to the explicit distribution provided, and if this description is of a distribution inside the property. Moreover they also proved that for a distribution property that requires  $\Omega(t)$  samples to test, any *public-coin* proof system for this property must satisfy  $s \cdot c = \Omega(t)$ . As mentioned above, verifying the distance from uniformity or approximating the entropy of a distribution requires  $\tilde{\Omega}(N)$  samples, and so, every *AM* protocol that verifies this property must also satisfy  $s \cdot c = \tilde{\Omega}(N)$ . Our protocol for this problem achieves  $c \cdot s = \tilde{O}(N^{4/3})$ , and the question of whether there exists a more efficient *public-coin* proof system for this problem remains open.

**Obtaining approximate tags of elements in  $[N]$** 

The method through which our protocol allows the verifier to verify *any* label invariant distribution property is by having the verifier uniformly draw elements from  $[N]$ , and verifiably obtain an approximation of the probability of each element according to  $D$ , that is correct on average (we call this a *uniformly drawn approximate tagged sample*). Formally, for some accuracy parameter  $\sigma \in (0, 1)$ , and a tuple  $(z_i) \in [N]^s$ , we define:

► **Definition 2** ( $\sigma$ -approximate tags for  $(z_i)$  with respect to  $D$ ).  $\sigma$ -approximate tags for  $(z_i)$  with respect to  $D$  is a tuple  $(\pi_i)_{i \in [s]} \in [0, 1]^s$  that satisfies the following inequality:

$$\frac{1}{s} \sum_{i \in [s]} \left( 1 - \min \left\{ \frac{D(z_i)}{\pi_i}, \frac{\pi_i}{D(z_i)} \right\} \right) \leq \sigma \quad (1)$$

In other words, on *average*,  $\pi_i \in [1 \pm \sigma]D(z_i)$ . A uniformly drawn approximate tagged sample allows to approximate the probability histogram of a distribution, as explained in the following sections. Note that in [13] and [14] the authors obtain an approximate tagged sample drawn according to  $D$ , rather than from a uniformly drawn sample, and use it to approximate the probability histogram of  $D$ . Thus, upon obtaining the probability histogram, our approaches converge, and we follow these works to bridge the gap between obtaining a probability histogram of a distribution and the estimation of distance from a label-invariant property. Note that the main difficulty is obtaining the tagged sample, a task that without communication would've required  $\tilde{\Omega}(N)$  samples, and so, this paper will focus on this point.

Moreover, [13, 14] not only contain secret coins, but also rely on the fact that the verifier can send samples from  $D$  to the prover. In this work, we allow the verifier to only send random coins, not even samples from  $D$ . This choice is justified in Chiesa and Gur [5], and allows our protocol to utilize properties of public-coin protocols over other objects with different access models.

We also show that a uniformly drawn approximate tagged-sample can also be used to verify distribution properties that are *not* label-invariant. Specifically, we also show that for the well-studied problem of approximating the distance of  $D$  from an explicit distribution  $Q$ , an approximate tagged uniform sample is sufficient:

► **Theorem 3** (Tolerant Verification of Identity). *Given an explicit description of distribution  $Q$  over  $[N]$ , parameters  $0 < \varepsilon_c < \varepsilon_f <$ , and sample access to distribution  $D$  over domain  $[N]$ , there exists a 2-message public-coin protocol, with verifier sample complexity and communication complexity  $\tilde{O}(N^{2/3}) \cdot \text{poly}(\frac{1}{\varepsilon_f - \varepsilon_c})$  such that:*

- *If  $\Delta_{SD}(D, Q) \leq \varepsilon_c$ , the verifier accepts with high probability.*
- *If  $\Delta_{SD}(D, Q) \geq \varepsilon_f$ , the verifier rejects with high probability.*

**1.2 Further Related Works**

Interactive proof systems were introduced in the seminal work of Goldwasser, Micali and Rackoff [10] in the context of proving computational statements about an input that is fully known to the prover and the verifier. In our work, the distribution can be thought of as the input, but it is not fully known to the verifier, and is accessed implicitly through samples. We aim for verification without examining the distribution in its entirety, using minimal resources (samples, communication, runtime, etc.).

Our work builds on a line of work that studied the power of sublinear time verifiers, who cannot read the entire input [6, 18, 12], on verifying properties of distributions using a small number of samples [5, 13, 14], and the rich literature of distribution testing, of which

most notably, we extensively use the ideas of Batu and Canonne in [3], as explained in the technical overview. We also note that Herman and Rothblum [15] recently showed that a very rich family of distribution properties, those that can be decided by a *small* circuit from an explicit description of the distribution, can be doubly-efficiently verified with a *secret-coin* protocol.

## 2 Technical Overview

As discussed in the introduction above, the protocol behind Theorem 1 is based on obtaining verified  $\Theta(\rho)$ -approximate tags with respect to  $D$  for a sample uniformly drawn from  $[N]$ . In this section, we describe the public-coin protocol for obtaining this object. We then detail how this tagged sample can be leveraged to verify membership in label-invariant distribution properties.

► **Theorem 4 (Informal).** *There exists a 2-message public-coin interactive protocol between a verifier and a (potentially malicious) prover, where the verifier receives as input parameters  $\sigma \in (0, 0.1)$  and  $N \in \mathbb{N}$ , as well as sample access to a distribution  $D$  over domain  $[N]$ . The communication complexity, verifier sample complexity, and verifier runtime are all  $s = \tilde{O}(N^{2/3}) \text{poly}(\sigma^{-1})$ , the honest prover with the same input as the verifier has sample complexity and runtime  $\tilde{O}(N) \text{poly}(\sigma^{-1})$ . At the end of the interaction, the verifier rejects or outputs  $(S_i) \in [N]^s$  that is drawn uniformly from  $[N]$ , and  $(\pi_i) \in [0, 1]^s$  such that:*

- *If the prover is honest, for all  $i \in [s]$ ,  $\pi_i = D(S_i)$ , and with probability at least 0.75, the verifier doesn't reject.*
- *Whatever strategy a dishonest prover follows, with probability at most 0.25 over the verifier's coin tosses and samples, the verifier accepts and outputs  $(\pi_i)$  such that doesn't satisfy Inequality (1).*

We outline the protocol behind Theorem 4. We highlight that some details are swept under the rug for sake of simplicity. In particular, we assume that  $D(x) \leq \frac{1}{s}$  for all  $x \in [N]$ . After we present the protocol under this assumption, we discuss how to remove this assumption.

### The communication phase

The verifier draws an i.i.d. sample  $S = (S_i)$  of size  $s = \tilde{O}(N^{2/3}) \cdot \text{poly}(\sigma^{-1})$  uniformly from  $[N]$ , and sends the sample to the prover. For each sample  $S_i$  received, the prover replies with  $\pi_i$  such that  $\pi_i = D(S_i)$ . Note that with high probability, due to the choice of  $s$ , there doesn't exist an element in  $x \in [N]$  that was sampled more than 3 times,<sup>2</sup> and in general, the fraction of elements that were sampled twice or three times is very small with respect to  $s$ . Therefore, for sake of simplicity, assume that  $S$  contains only unique elements.

Moreover, since we assumed  $D(x) \leq \frac{1}{s}$  for all  $x \in [N]$ , by choice of  $s$ , the sample  $S$  contains with overwhelming probability *many* samples uniformly distributed inside  $\text{Supp}(D)$ .

### Verifying the prover's message

The verifier divides the samples in  $S$  into *buckets* according to their alleged probability, where inside each bucket all the samples are claimed to have roughly the same mass. Concretely, for  $\tau = O(\sigma^3)$ , and for every  $j$ , denote by  $B_j^S \subseteq [s]$  the collection of indices in  $S$  that the

<sup>2</sup> The probability that 4 samples collide is  $\sum_{x \in [N]} D(x)^4 = \frac{1}{N^3}$  while there are only  $\binom{s}{4} = O(N^{8/3})$  possible 4-tuples in the sample  $S$ .

prover claimed have probability in the range  $\left[\frac{(1+\tau)^j}{N}, \frac{(1+\tau)^{j+1}}{N}\right]$ . The verifier then tests for every such  $j$  that the *average probability* of the elements in  $B_j^S$  is indeed roughly  $\frac{(1+\tau)^j}{N}$ , and that  $D|_{B_j^S}$  is *close to uniform*:

- **Checking that the average mass is correct.** The verifier draws a fresh sample  $T$ , and checks that the empirical mass of  $B_j^S$  in  $T$  is roughly  $s \cdot |B_j^S| \cdot \frac{(1+\tau)^j}{N}$ , and rejects otherwise. Observe that for any distribution  $D$ , the true mass of  $B_j^S$  is  $\sum_{k \in B_j^S} D(S_k)$ . And so, by choice of  $s$ , since the empirical mass of  $B_j^S$  in  $T$  is strongly concentrated around its mean, if the test passes, then with high probability:

$$s \cdot \sum_{k \in B_j^S} D(x) \stackrel{\tau}{\approx} s \cdot |B_j^S| \cdot \frac{(1+\tau)^j}{N}$$

Where for  $\alpha \in (0, 1)$  we use the notation  $a \stackrel{\alpha}{\approx} b$  to indicate that  $a \in (1 \pm \alpha)b$ . We conclude that with high probability:

$$\mathbb{E}_{k \sim_{uni} B_j^S} [D(S_k)] \stackrel{O(\tau)}{\approx} \frac{(1+\tau)^j}{N} \quad (2)$$

- **Verifying that  $D|_{B_j^S}$  is close to uniform.** The verifier draws another fresh  $D$ -sample  $T'$  of size  $s$ , and counts how many *3-way collisions* occur between elements in  $B_j^S$  and the two samples  $T, T'$ , i.e. the number of 3-tuples  $(k, r, r') \in [s]^3$  satisfy  $k \in B_j^S$ ,  $S_k = T_r = T'_{r'}$ . If this quantity is far from  $s^2 \cdot |B_j^S| \cdot \left(\frac{(1+\tau)^j}{N}\right)^2$ , the verifier rejects. Similar to before, for any fixed pair of entries in  $T, T'$ ,  $(r, r') \in [s]^2$ , the true expected number of  $k \in B_j^S$  for which  $S_k = T_r = T'_{r'}$  is  $\sum_{k \in B_j^S} (D(S_k))^2$ . The total expected number of such 3-tuples is  $s^2 \cdot \sum_{k \in B_j^S} (D(S_k))^2$ . This quantity is also strongly concentrated around its mean by choice of  $s = \Theta(N^{2/3})\text{poly}(\sigma^{-1})$ . We conclude that if this test passed, then with high probability:

$$s^2 \cdot \sum_{k \in B_j^S} (D(S_k))^2 \stackrel{O(\tau)}{\approx} s^2 |B_j^S| \cdot \left(\frac{(1+\tau)^j}{N}\right)^2$$

And equivalently:

$$\mathbb{E}_{k \sim_{uni} B_j^S} [(D(S_k))^2] \stackrel{O(\tau)}{\approx} \left(\frac{(1+\tau)^j}{N}\right)^2 \quad (3)$$

We are thus left to argue that Equations (2) and (3) imply that  $D|_{B_j^S}$  is close to uniform. Following Batu and Canonne [3], observe that:

$$\text{Var}_{k \sim_{uni} B_j^S} [D(S_k)] = \mathbb{E}_{k \sim_{uni} B_j^S} [(D(S_k))^2] - \left(\mathbb{E}_{k \sim_{uni} B_j^S} [D(S_k)]\right)^2$$

And so, assuming Equations (2) and (3) hold, we get that  $\text{Var}_{k \sim_{uni} B_j^S} [D(S_k)] = O(\tau) (\mathbb{E}[D(x)])^2$ . Using Chebychev's Inequality:

$$\Pr_{k \sim_{uni} B_j^S} \left( \left| D(S_k) - \mathbb{E}_{k \sim_{uni} B_j^S} [D(S_k)] \right| \geq O\left(\sqrt{\frac{\tau}{\sigma}}\right) \cdot \mathbb{E}_{k \sim_{uni} B_j^S} [D(S_k)] \right) \leq O(\sigma) \quad (4)$$



From which we conclude all but  $\sigma$ -fraction of entries  $i \in B_j^S$  satisfy:

$$\pi_i \stackrel{O(\tau)}{\approx} \frac{(1+\tau)^j}{N} \stackrel{O(\tau)}{\approx} \mathbb{E}_{k \sim^{uni} B_j^S} [D(S_k)] \stackrel{O(\sqrt{\tau/\sigma})}{\approx} D(S_i)$$

Where the first inequality stems from the definition of  $B_j^S$ , the second from Equation (2), and the last from Inequality (4). Plugging in  $\tau = O(\sigma^3)$ , we get:  $\pi_i \stackrel{O(\sigma)}{\approx} D(S_i)$ .

We thus showed that if both verifier tests pass, then with high probability over the randomness of the verifier, it holds that for every  $j$ , the tags over  $B_j^S$  are  $\sigma$ -approximately correct, from which Inequality (1) is inferred.

### Assuming $D$ contains no heavy elements

Observe that the probability of all elements with probability larger than  $1/s$  can be well-approximated through their empirical mass in a sample of size  $\tilde{\Theta}(s)$  from  $D$ . Therefore, we can think of a verifier that estimates without need of a prover the mass of all such elements. This process is described in detail in [13], and we describe it shortly here. The reader is referred to their work for further detail. After receiving the prover's tags, the verifier performs the following step: the verifier draws a fresh  $D$ -sample, denoted  $\mathcal{H}$ , of size  $\tilde{O}(s)\text{poly}(\sigma^{-1})$  from  $D$ . With high probability, by a *coupon-collector* argument, this set contains all elements with probability at least  $\frac{1}{s}$  (if any exist).

The verifier tests the mass of  $\mathcal{H}$  by drawing a fresh sample and examining the empirical mass of  $\mathcal{H}$  in that new sample. If it is significant, i.e.  $\Omega(\sigma)$ , the verifier “learns”  $D|_{\mathcal{H}}$  up to  $\sigma$  distance by subsampling from this distribution and running a *folklore distribution learner* (see Theorem 4). This requires  $\tilde{O}(s)\text{poly}(\sigma^{-1})$  samples from  $D$ , and thus doesn't incur significant overhead to the sample complexity of the protocol. Thus, the verifier obtains an explicit description of the distribution  $P_{\mathcal{H}}$ , which is  $O(\sigma)$ -close to  $D|_{\mathcal{H}}$ . Since  $\mathcal{H}$  is a set of size at most  $s$ , and the sample  $S$  was drawn drawn i.i.d. from  $[N]$ , with overwhelming probability it holds that  $|S \cap \mathcal{H}| = O(N^{1/3}) = o(s)$ , and in order to verify the prover's answer's in the protocol described above, the verifier can just “erase” every element in  $S$  that appeared in  $\mathcal{H}$ , and run the protocol presented above over just elements guaranteed with high probability to be of probability at most  $1/s$ , without affecting the correctness of the protocol. Thus, the verifier obtains full tags for  $\mathcal{H}$ , and tags for  $S \setminus \mathcal{H}$ . Later, the verifier can “fill-in” the missing parts in  $S$  to obtain a full tagged sample. If  $D$  is entirely supported over heavy elements, then the protocol can be avoided all together by also checking the mass of  $\mathcal{H}$  is larger than  $1 - O(\sigma)$ , and ignoring the prover's message.

### Verifying label-invariant distribution properties

In order to verify *label-invariant* distribution properties, it suffices to know the *probability histogram* of the distribution, i.e., how many elements have probability  $p$  for every  $p \in [0, 1]$ . Herman and Rothblum [13] observed that for many natural properties an approximation of this histogram is sufficient, and define the  $\tau$ -bucket histogram as follows:

► **Definition 5** ( $\tau$ -bucket histogram of  $D$ ). For any  $j \in \{\dots, -1, 0, 1, \dots, \frac{\log N}{\tau}\}$ , the  $j$ 'th bucket of  $D$  over domain  $[N]$  is:

$$B_j^D = \left\{ x \in \text{Supp}(D) : D(x) \in \left[ \frac{(1+\tau)^j}{N}, \frac{(1+\tau)^{j+1}}{N} \right) \right\}$$

The  $\tau$ -bucket histogram of  $D$  is the tuple  $((j, D(B_j^D)))_{j: B_j^D \neq \emptyset}$ .

In [13] the authors focus their attention on those label-invariant distribution properties for which the information  $((j, D(B_j^D)))_{j: B_j^D \neq \emptyset}$  is sufficient in order to efficiently approximate the *distance* (in total-variation) of  $D$  from the property. They say that such properties admit an *efficient approximate decision procedure*, and show that many natural label-invariant problems are of this type, including the property of having Shannon entropy roughly  $k$ , or being close to uniform over some set of size  $M \leq N$ .

In our protocol the verifier obtains a uniformly drawn tagged sample<sup>3</sup>. We argue that this tagged sample allows the verifier to compute an approximation of the bucket histogram of  $D$ : if our protocol didn't end in rejection, then with high probability, the tags are roughly correct. In other words, for every  $j$ ,  $\frac{|B_j^S|}{s}$  is the empirical mass of  $B_j^D$  in the uniform sample  $S$ . Since we expect there to be about  $\frac{|B_j^D|}{N}$ -fraction of samples in  $S$  that landed in  $B_j^D$ , we conclude that:

$$\frac{|B_j^S|}{s} \approx \frac{|B_j^D|}{N}$$

And since  $D(B_j^D) \approx |B_j^D| \cdot \frac{(1+\tau)^j}{N}$ , if we set  $p_j = \left(\frac{|B_j^S|}{s} \cdot N\right) \cdot \frac{(1+\tau)^j}{N}$ , then  $D(B_j^D) \approx p_j$ , and we get with high probability, a  $\tau$ -histogram which is  $O(\sigma)$  close to the true histogram of  $D$  in the following sense: there exists a distribution  $D'$  with histogram exactly  $((j, p_j))$  that is  $O(\sigma)$ -close to  $D$  in total variation distance. Thus, using the decision procedure, the verifier decides whether  $((j, p_j))$  is consistent with some distribution close to  $\mathcal{P}$ , and thus, conclude whether  $D$  is far from the property, or close to it.

### 3 Preliminaries

For an integer  $n \in \mathbb{N}$ , we use  $[n]$  to denote the set  $\{1, \dots, n\}$ .

► **Definition 6.** *The total variation distance (alt. statistical distance) between distributions  $P$  and  $Q$  over a finite domain  $X$  is defined as:*

$$\Delta_{SD}(P, Q) = \frac{1}{2} \sum_{x \in X} |P(x) - Q(x)|$$

► **Theorem 7** (Folklore distribution learner [8]). *There exists an algorithm that given sample access to a distribution  $P$  over the domain  $[N]$ , and an accuracy parameter  $\alpha \in (0, 1)$ , it runs in time  $\tilde{O}(N/\alpha^2)$ , takes  $O(N/\alpha^2)$  samples, and with probability at least 0.99 outputs a full description of a distribution  $P_{approx}$  such that  $\Delta_{SD}(P, P_{approx}) \leq \alpha$ .*

► **Definition 8** (Distribution property). *We say the  $\mathcal{P} = (\mathcal{P}_N)_{N \in \mathbb{N}}$  is a distribution property if  $\mathcal{P}_N \subseteq \Delta_N$ , where  $\Delta_N$  is the set of all distributions over domain  $[N]$ .*

► **Definition 9** (Distribution tester for property  $\mathcal{P}$ ). *Let  $\mathcal{P}$  be a distribution property. A tester  $T$  of property  $\mathcal{P}$  is a probabilistic oracle machine, that on input parameters  $N$  and  $\varepsilon$ , and oracle access to a sampling device for a distribution  $D$  over a domain of size  $[N]$ , outputs a binary verdict that satisfies the following two conditions:*

1. *If  $D \in \mathcal{P}_N$ , then  $\Pr(T^D(N, \varepsilon) = 1) \geq 2/3$ .*
2. *If  $\Delta_{SD}(D, \mathcal{P}_N) > \varepsilon$ , then  $\Pr(T^D(N, \varepsilon) = 0) \geq 2/3$ .*

<sup>3</sup> Here we differ from [13] that obtain a  $D$ -sampled tagged sample, i.e.  $(z_i)$  in their case was drawn from  $D$ .

In the context of this work, the relevant distance measure is *statistical distance* as defined above. An extension of this definition, introduced by Parnas, Ron, and Rubinfeld [16] is the following:

► **Definition 10** ( $(\varepsilon_c, \varepsilon_f)$ -tolerant distribution property tester). *For parameters  $\varepsilon_c, \varepsilon_f \in [0, 1]$  such that  $\varepsilon_c < \varepsilon_f$ , a  $(\varepsilon_c, \varepsilon_f)$ -tolerant tester  $T$  of property  $\Pi$  is a probabilistic oracle machine, that on inputs  $N, \varepsilon_c, \varepsilon_f$  and given oracle access to a sampling device for distribution  $D$  over a domain of size  $N$ , outputs a binary verdict that satisfies the following two conditions:*

1. *If  $\delta(D, \Pi_N) \leq \varepsilon_c$ , then  $\Pr(T^D(N, \varepsilon_c, \varepsilon_f) = 1) \geq 2/3$ .*
2. *If  $\delta(D, \Pi_N) \geq \varepsilon_f$ , then  $\Pr(T^D(N, \varepsilon_c, \varepsilon_f) = 0) \geq 2/3$ .*

Note that a tolerant distribution test is for some property  $\Pi$  is at least as hard as a standard non-tolerant tester for the same property.

► **Definition 11** (Proof system for tolerant distribution testing problems). *A proof system for a tolerant distribution testing problem  $\mathcal{P}$  with parameters  $\varepsilon_c$  and  $\varepsilon_f$  is a two-party game, between a verifier executing a probabilistic polynomial time strategy  $V$ , and a prover that executes a strategy  $P$ . Given that both  $V$  and  $P$  have black-box sample access to distribution  $D$  over the domain  $[N]$ , and are given  $N$ , the interaction should satisfy the following conditions:*

- **Completeness:** *For every  $D$  over domain of size at most  $N$ , such that  $\Delta_{SD}(D, \mathcal{P}_N) \leq \varepsilon_c$ , the verifier  $V$ , after interacting with the prover  $P$ , accepts with probability at least  $2/3$ .*
- **Soundness:** *For every  $D$  over domain of size at most  $N$  such that  $\Delta_{SD}(D, \mathcal{P}_N) \geq \varepsilon_f$ , and every cheating strategy  $P^*$ , the verifier  $V$ , after interacting with the prover  $P^*$ , rejects with probability at least  $2/3$ .*

The complexity measures associated with the protocol are: the sample complexity of the verifier as the honest prover (strategy  $P$ ), the communication complexity, the runtime of both agents, and the round complexity (how many messages were exchanged).

► **Definition 12** (Label invariant distribution property). *A distribution property  $\mathcal{P}$  is called label invariant if for all  $N \in \mathbb{N}$ , it holds that any permutation  $\sigma$  over  $N$  elements satisfies that  $D \in \mathcal{P}_N$  if and only if  $\sigma(D) \in \mathcal{P}_N$ .*

## 4 Public Coin Protocol for Verified Tagged Sample

Using the same approach as Herman and Rothblum [13], we provide an algorithm to obtain a *tagged sample* assuming that the samplable distribution  $D$  satisfies that for every  $x \in [N]$ ,  $D(x) \leq \frac{1}{s}$ , where  $s = O\left(\frac{\log N}{\varepsilon^5} \cdot N^{2/3}\right)$ . In Section 2 we discuss why we can assume this without loss of generality.

► **Theorem 13.** *There exists 2-message AM interactive protocol between an honest verifier and a (potentially malicious) prover, where the verifier receives as input parameters  $\sigma \in (0, 0.1)$  and  $100 < N \in \mathbb{N}$ , as well as sample access to a distribution  $D$  over domain  $[N]$ . Set  $\tau = \frac{\sigma^3}{8000}$ . Assume  $D(x) \leq \frac{1}{s}$  for  $s = O\left(\frac{\log N}{\varepsilon^5} \cdot N^{2/3}\right)$ . The communication complexity, verifier sample complexity, and verifier runtime are all  $s$ . Given sample access to the distribution  $D$ , the honest prover requires with high probability  $\tilde{O}(N)$   $\text{poly}(\sigma^{-1})$  samples and runtime.*

*At the end of the interaction, the verifier rejects or outputs  $((z_i, \pi_i))_{i \in [s]}$  where  $(z_i)_{i \in [s]}$  is a sample of size  $s$  drawn uniformly i.i.d. from  $[N]$  and:*

- **Completeness.** *If the prover is honest, then with probability at least 0.75, the verifier doesn't reject, and  $((z_i, \pi_i))_{i \in [s]}$  satisfies  $\frac{1}{s} \sum_{i \in [s]: \pi_i \geq \frac{\sigma}{1000N}} \left(1 - \min\left\{\frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i}\right\}\right) = O(\tau)$ , while  $\frac{1}{s} \sum_{i \in [s]: \pi_i \leq \frac{\sigma}{1000N}} D(z_i) \leq \frac{\sigma}{50N}$ .*

## 72:10 Public Coin Interactive Proofs for Label-Invariant Distribution Properties

- **Soundness.** *Whatever strategy a dishonest prover follows, with probability at most 0.25 over the verifier's coin tosses and samples, they accept and  $((z_i, \pi_i))_{i \in [s]}$  satisfies:*

$$\frac{1}{s} \sum_{i \in [s]: \pi_i \geq \frac{\sigma}{1000N}} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \geq \sigma \quad (5)$$

or

$$\frac{1}{s} \sum_{i \in [s]: \pi_i \leq \frac{\sigma}{1000N}} D(z_i) \geq \frac{\sigma}{10N} \quad (6)$$

Note that we use the convention that  $\min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} = 1$  if  $\pi_i = 0$  and  $D(z_i) \neq 0$ , or  $\pi_i \neq 0$  and  $D(z_i) = 0$ .

We show that Protocol 1 satisfies the conditions of Theorem 13.

### ■ Protocol 1 Public-Sample Tagged Sample Retrieval Protocol.

**Input:** parameters  $N \in \mathbb{N}$ ,  $\sigma \in (0, 1)$ , as well as sample access to distribution  $D$  over domain  $[N]$  such that for all  $x \in [N]$ ,  $D(x) \leq \frac{1}{s}$  for  $s = O\left(\frac{\log N}{\epsilon^5} N^{2/3}\right)$ .

1. V: draw  $s$  uniformly from  $[N]$ . Denote the sample  $(S_i)_{i \in [s]}$ . Reject if there exists  $x \in [N]$  such that  $x$  appears more than  $\log N$  times in  $S$ . Otherwise, send  $(S_i)$  to P.
2. P: set  $\tau = \frac{\sigma^3}{80000}$ . For every  $i \in [s]$ , if  $D(S_i) \geq \frac{\sigma}{100N}$ , send  $\pi_i$  such that  $\pi_i = D(S_i)$ , otherwise, send  $\pi_i = 0$ .
3. V: for every  $j$  set  $S^j = \left\{ i \in [s] : \pi_i \in \left[ \frac{e^{j\tau}}{N}, \frac{e^{(j+1)\tau}}{N} \right) \right\}$ . Draw two fresh samples of size  $s$  from  $D$ ,  $T = (T_i)_{i \in [s]}$  and  $T' = (T'_i)_{i \in [s]}$ . For every  $j$  such that  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\epsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , set:

$$\tilde{C}_j^{pair} = \left| \left\{ (k, r) \in [s]^2 : k \in S^j, S_k = T_r \right\} \right|$$

$$\tilde{C}_j^{triple} = \left| \left\{ (k, r, r') \in [s]^3 : k \in S^j, S_k = T_r = T'_{r'} \right\} \right|$$

Reject unless for all such  $j$ :

$$\left| \tilde{C}_j^{pair} - s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} \right| \leq 4\tau \cdot s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} \quad (7)$$

And

$$\left| \tilde{C}_j^{triple} - s^2 \cdot |S^j| \cdot \left( \frac{e^{j\tau}}{N} \right)^2 \right| \leq 4\tau \cdot s^2 \cdot |S^j| \cdot \left( \frac{e^{j\tau}}{N} \right)^2 \quad (8)$$

4. V: denote  $S^{-\infty} = \{i \in [s] : \pi_i = 0\}$ . Reject unless  $\tilde{C}_{-\infty}^{pair} \leq s \cdot |S^{-\infty}| \cdot \frac{\sigma}{50N}$ .
5. V: Output  $((S_i, \pi_i))_{i \in [s]}$

### 4.1 Protocol 1 is Complete

We first show that Step 1 of Protocol 1 does not result in rejection.

▷ **Claim 14.** With probability at least 0.99 over the choice of  $S$ , there doesn't exist an element  $x \in [N]$  that was sampled more than 3 times in  $S$ , and the verifier doesn't reject after Step 1 of Protocol 1.

Proof. Fix  $x \in [N]$  and  $i_1, i_2, i_3, i_4 \in [s]$  such that for all  $k, k' \in [\log N]$ ,  $i_k \neq i_{k'}$ . Note that:

$$\Pr_S(S_{i_1} = S_{i_2} = S_{i_3} = S_{i_4}) = \left(\frac{1}{N}\right)^4$$

There are  $\binom{s}{4}$  possible choices for  $i_1, i_2, i_3, i_4 \in [s]$ . Therefore, the probability that there exists some set of 4 indices whose respective samples equal  $x$  is at most:

$$\binom{s}{4} \cdot \frac{1}{N^4} \leq \left(\frac{s}{N}\right)^4 \leq \frac{1}{N^{4/3}}$$

Taking the union bound over all possible  $x \in [N]$  yields the desired result. ◁

Next, we argue that if the prover is honest, with high probability, the verifier collision tests don't result in rejection.

▷ **Claim 15.** Assuming the verifier didn't reject after Step 1 and that the prover is honest, then with probability at least 0.8 over the choice of  $T, T'$  the verifier doesn't reject.

Proof. For every  $j$  such that  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$  and  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  by Propositions 30 and 31 and choice of  $s$ , it also holds that:

$$\begin{aligned} \mathbb{E} \left[ \tilde{C}_j^{pair} \right] &= s \left( \sum_{i \in S^j} D(S_i) \right) \geq s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} \geq \frac{300 \log^2 N}{\tau^3} \\ \mathbb{E} \left[ \tilde{C}_j^{triple} \right] &\geq s^2 \sum_{i \in S^j} (D(S_i))^2 = s^2 \cdot |S^j| \cdot \left( \frac{e^{j\tau}}{N} \right)^3 \geq \frac{300 \log^2 N}{\tau^3} \end{aligned}$$

And so, since there are at most  $2 \log N / \tau$  buckets for which  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , we conclude from Propositions 30 and 31 that with probability at least 0.8 over the choice of  $T, T'$  for all  $j$  as described in statement it holds that:

$$\left| \tilde{C}_j^{triple} - \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{triple} \right]}} \leq 4\tau s^2 |S^j| \left( \frac{e^{j\tau}}{N} \right)^2$$

And similarly:

$$\left| \tilde{C}_j^{pair} - \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{pair} \right]}} \leq (e^\tau - 1) s |S^j| \frac{e^{j\tau}}{N} \cdot \tau \leq 4\tau s |S^j| \frac{e^{j\tau}}{N}$$

◁

▷ **Claim 16.** If the prover is honest, with high probability over  $T$ , the final verifier test passes with high probability, and:

$$\frac{1}{s} \sum_{i \in [s]: \pi_i < \frac{\sigma}{1000N}} D(S_i) \leq \frac{\sigma}{10N} \tag{9}$$

## 72:12 Public Coin Interactive Proofs for Label-Invariant Distribution Properties

Proof. Since the prover is honest,  $\mathbb{E} [\tilde{C}_{-\infty}] = s \cdot \sum_{i \in S^{-\infty}} D(S_i) \leq s \cdot |S^{-\infty}| \cdot \frac{\sigma}{1000N}$ , and so, by Markov's Inequality, with probability at least 0.95,  $\tilde{C}_{-\infty} \leq s \cdot |S^{-\infty}| \cdot \frac{\sigma}{50N}$ , and the final test passes. Moreover, Inequality (9) holds.  $\triangleleft$

► **Remark 17 (Honest prover complexity).** For sake of simplicity we assume the honest prover in Protocol 1 knows  $D(S_i)$  exactly. However, this is not necessary. A prover that approximates this quantity for every sample up to sufficient accuracy using only  $\tilde{O}(N)\text{poly}(\tau^{-1})$  samples suffices. See Remark 4.14 in [14] for a detailed discussion.

### 4.2 Protocol 1 is Sound

Note that by Claim 14, regardless of the prover's response, the verifier rejects after Step 1 with probability at most 0.01, and so, throughout this section, we assume that Step 1 passed, and  $S$  doesn't contain elements appearing more than 4 times, even when not stated explicitly.

First, we address the *last* verifier test:

► **Claim 18.** For every index  $j$  such that  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$  and  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$ , with probability at least 0.98 over the choice of  $T$  and  $T'$ , either the verifier rejects, or it holds that:

$$\mathbb{E} [\tilde{C}_j^{pair}] \geq \frac{300 \log^2 N}{\tau^3} \quad (10)$$

and

$$\mathbb{E} [\tilde{C}_j^{triple}] \geq \frac{300 \log^2 N}{\tau^3} \quad (11)$$

Proof. Fix some  $j_0$  such that  $|S^{j_0}| \geq s \cdot \frac{\varepsilon \tau}{100 \log N}$ ,  $\frac{e^{j_0 \tau}}{N} \geq \frac{\varepsilon}{100N}$ , and also  $\mathbb{E} [\tilde{C}_{j_0}^{triple}] < \frac{300 \log^2 N}{\tau^3}$ . By Markov's Inequality, with probability at least 0.99:

$$\tilde{C}_{j_0}^{triple} \leq 100 \mathbb{E} [\tilde{C}_{j_0}^{triple}] \leq \frac{30000 \log^2 N}{\tau^3}$$

However, the verifier rejects unless:

$$\tilde{C}_{j_0}^{triple} \geq (1 - 4\tau) s^2 |S^{j_0}| \left( \frac{e^{j_0 \tau}}{N} \right)^2 \geq s^3 \cdot \frac{\tau \varepsilon^3}{2 \cdot 100^3 N^3 \log N} > \frac{30000 \log^2 N}{\tau^3}$$

Where the last inequality is justified since  $s \geq \frac{300 \log N}{\tau^{4/3 \varepsilon}} N^{2/3}$ . We thus conclude that for every  $j$  such that  $v_{j_0} \geq \frac{\varepsilon \tau}{100 \log N}$ ,  $\frac{e^{j_0 \tau}}{N} \geq \frac{\varepsilon}{100N}$ , either  $\mathbb{E} [\tilde{C}_j^{triple}] \geq \frac{300 \log^2 N}{\tau^3}$  or the verifier reject with probability at least 0.99. An analogous argument can be made w.r.t. to  $\tilde{C}_j^{pair}$ . Taking the union bound over both these events yields the required result.  $\triangleleft$

► **Claim 19.** With probability at least 0.8 over the choice of  $T$  and  $T'$ , for every  $j$  such that  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , and for which Inequalities (10) and (11) hold, it further holds that:

$$\left| \tilde{C}_j^{pair} - s \sum_{i \in S^j} D(S_i) \right| \leq 4\tau \cdot s \sum_{i \in S^j} D(S_i) \quad (12)$$

As well as:

$$\left| \tilde{C}_j^{triple} - s^2 \sum_{i \in S^j} (D(S_i))^2 \right| \leq 4\tau \cdot s^2 \sum_{i \in S^j} (D(S_i))^2 \quad (13)$$

Proof. By Propositions 30 and 31 it holds that with probability 0.8 over the choice of  $T$  and  $T'$  for every  $j$  such that  $|S^j| \geq s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\varepsilon}{100N}$ , the following holds:

$$\left| \tilde{C}_j^{pair} - \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{pair} \right]}}$$

$$\left| \tilde{C}_j^{triple} - \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{triple} \right]}}$$

Moreover, from the same propositions we know that:

$$\mathbb{E} \left[ \tilde{C}_j^{pair} \right] = s \sum_{i \in S^j} D(S_i)$$

$$\mathbb{E} \left[ \tilde{C}_j^{triple} \right] = s^2 \sum_{i \in S^j} (D(S_i))^2$$

We thus conclude that for all the  $j$  as specified above:

$$\left| \tilde{C}_j^{pair} - s \sum_{i \in S^j} D(S_i) \right| \leq \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{pair} \right]}} \leq \tau s \sum_{i \in S^j} D(S_i)$$

Where the last inequality above stems from the assumption that Inequality (11) holds. Similarly:

$$\left| \tilde{C}_j^{triple} - s^2 \sum_{i \in S^j} (D(S_i))^2 \right| \leq \tau s^2 \sum_{i \in S^j} (D(S_i))^2 \quad \triangleleft$$

▷ **Claim 20.** Assuming the verifier didn't reject, with probability at least 0.8 over the choice of  $T$  and  $T'$ , for every  $j$  such that  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , and for which Inequalities (10) and (11) hold. It further holds that:

$$\frac{1}{|S^j|} \sum_{i \in S^j} D(S_i) \in \frac{e^{j\tau}}{N} [1 - 10\tau, 1 + 10\tau] \quad (14)$$

$$\frac{1}{|S^j|} \sum_{i \in S^j} (D(S_i))^2 \in \left( \frac{e^{j\tau}}{N} \right)^2 [1 - 10\tau, 1 + 10\tau] \quad (15)$$

Proof. By Claim 19, with probability at least 0.8 over the choice of  $T$  and  $T'$ , for every  $j$  such that and  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , and for which Inequalities (10) and (11) hold, Inequalities (12) and (13) hold.

Furthermore, if the verifier didn't reject, for all such  $j$ , Inequalities (7) and (8) holds as well for all such  $j$ . Putting it all together, we get that:

$$\left| s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} - s \sum_{i \in S^j} D(S_i) \right| \leq \left| s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} - \tilde{C}_j^{pair} \right| + \left| \tilde{C}_j^{pair} - s \sum_{i \in S^j} D(S_i) \right| \quad (16)$$

$$\leq 4\tau s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} + 4\tau s \sum_{i \in S^j} D(S_i) \quad (17)$$

Rearranging Inequality (16):

$$s \sum_{i \in S^j} D(S_i) \in s \cdot |S^j| \cdot \frac{e^{j\tau}}{N} \left[ \frac{1-4\tau}{1+4\tau}, \frac{1+4\tau}{1-4\tau} \right]$$

Likewise:

$$\left| s^2 \cdot |S^j| \left( \frac{e^{j\tau}}{N} \right)^2 - s^2 \sum_{i \in S^j} (D(S_i))^2 \right| \leq +4\tau s^2 \cdot |S^j| \left( \frac{e^{j\tau}}{N} \right)^2 + 4\tau s^2 \sum_{i \in S^j} (D(S_i))^2 \quad (18)$$

Similarly, for Inequality (18):

$$s^2 \sum_{i \in S^j} (D(S_i))^2 \in s^2 \cdot |S^j| \left( \frac{e^{j\tau}}{N} \right)^2 \left[ \frac{1-4\tau}{1+4\tau}, \frac{1+4\tau}{1-4\tau} \right]$$

And through the relation  $\frac{1-4\tau}{1+4\tau} \geq 1 - 10\tau$  and  $\frac{1+4\tau}{1-4\tau} \leq 1 + 10\tau$  that holds for all  $\tau > 0$ , we get the desired result.  $\triangleleft$

► **Definition 21.** Define the distribution  $U_{S^j}$  to be the uniform distribution over  $S^j$ .

▷ **Claim 22.** Assuming the verifier didn't reject, with probability at least 0.8 over the choice of  $T$  and  $T'$ , for every  $j$  such that  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , and for which Inequalities (10) and (11) hold. It further holds that:

$$\mathbb{E}_{i \sim U_{S^j}} [D(S_i)] \in \frac{e^{j\tau}}{N} [1 - 10\tau, 1 + 10\tau]$$

$$\text{Var}_{i \sim U_{S^j}} [D(S_i)] \leq 60\tau (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2$$

Proof. With high probability, for all  $j$  as specified in the claim statement, by Claim 20:

$$\mathbb{E}_{i \sim U_{S^j}} [D(S_i)] = \sum_{i \in S_i} \frac{1}{|S^j|} D(S_i) \in \frac{e^{j\tau}}{N} \cdot [1 - 10\tau, 1 + 10\tau]$$

Furthermore:

$$\mathbb{E}_{i \sim U_{S^j}} \left[ (D(S_i))^2 \right] = \frac{1}{|S^j|} \sum_{i \in S^j} (D(S_i))^2 \quad (19)$$

$$\leq (1 + 10\tau) \left( \frac{e^{j\tau}}{N} \right)^2 \quad (20)$$

$$\leq (1 + 10\tau) (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 \frac{1}{(1 - 10\tau)^2} \quad (21)$$

$$\leq (1 + 40\tau) (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 \quad (22)$$

And so, we conclude that:

$$\begin{aligned} \text{Var}_{i \sim U_{S^j}} [D(S_i)] &= \mathbb{E}_{i \sim U_{S^j}} \left[ (D(S_i))^2 \right] - (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 \\ &\leq (1 + 40\tau) (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 - (1 - 20\tau) (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 \\ &\leq 60\tau (\mathbb{E}_{i \sim U_{S^j}} [D(S_i)])^2 \end{aligned} \quad \triangleleft$$



▷ Claim 23. Assuming the verifier didn't reject, with probability at least 0.8 over the choice of  $T$  and  $T'$ , for every  $j$  such that  $|S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N}$ , and for which Inequalities (10) and (11) hold, it further holds that:

$$\mathbb{E}_{i \sim U_{S^j}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \geq 1 - \frac{\sigma}{50} \quad (23)$$

Proof. By Claim 22, for every  $j$  as specified in the claim statement, it holds that:

$$\mathbb{E}_{i \sim U_{S^j}} [D(S_i)] \in \frac{e^{j\tau}}{N} [1 - 10\tau, 1 + 10\tau]$$

$$\text{Var}_{i \sim U_{S^j}} [D(S_i)] \leq 60\tau \left( \mathbb{E}_{i \sim U_{S^j}} [D(S_i)] \right)^2$$

Therefore, through Chebychev's Inequality:

$$\Pr_{i \sim U_{S^j}} \left( |D(S_i) - \mathbb{E}[D(S_i)]| \geq \sqrt{\frac{6000\tau}{\sigma}} \cdot \mathbb{E}[D(S_i)] \right) \leq \frac{60\tau \left( \mathbb{E}_{i \sim U_{S^j}} [D(S_i)] \right)^2}{\left( \mathbb{E}_{i \sim U_{S^j}} [D(S_i)] \right)^2 \cdot 6000\tau/\sigma} \leq \frac{\sigma}{100}$$

Observe that with probability at least  $1 - \frac{\sigma}{100}$  over the choice of  $i \sim U_{S^j}$  it holds that:

$$\begin{aligned} |D(S_i) - \pi_i| &\leq |D(S_i) - \mathbb{E}[D(S_i)]| + \left| \mathbb{E}[D(S_i)] - \frac{e^{j\tau}}{N} \right| + \left| \frac{e^{j\tau}}{N} - \pi_i \right| \\ &\leq \sqrt{\frac{6000\tau}{\sigma}} \cdot \mathbb{E}[D(S_i)] + \left| \mathbb{E}[D(S_i)] - \frac{e^{j\tau}}{N} \right| + (e^\tau - 1) \cdot \frac{e^{j\tau}}{N} \\ &\leq \sqrt{\frac{6000\tau}{\sigma}} \cdot \frac{e^{j\tau}}{N} (1 + 10\tau) + 12\tau \cdot \frac{e^{j\tau}}{N} \\ &\leq \left( 2\sqrt{\frac{6000\tau}{\sigma}} + 12\tau \right) \frac{e^{j\tau}}{N} \\ &\leq e^\tau \left( 2\sqrt{\frac{6000\tau}{\sigma}} + 12\tau \right) \pi_i \\ &\leq \left( 3\sqrt{\frac{6000\tau}{\sigma}} + 12\tau \right) \pi_i \end{aligned}$$

Where the second to last inequality stems from the fact that by definition for all  $i \in S^j$ ,  $\pi_i \in \left[ \frac{e^{j\tau}}{N}, \frac{e^{(j+1)\tau}}{N} \right]$ . We conclude that for all such  $i$  it holds that:

$$\frac{D(S_i)}{\pi_i} \in \left[ 1 - 3\sqrt{\frac{6000\tau}{\sigma}} - 12\tau, 1 + 3\sqrt{\frac{6000\tau}{\sigma}} + 12\tau \right]$$

By choice of  $\tau$ , this implies that with probability at least  $1 - \frac{1}{100\sigma}$  over the choice of  $i \sim U_{S^j}$ , it holds that:

$$\frac{D(S_i)}{\pi_i} \in \left[ 1 - \frac{\sigma}{100}, 1 + \frac{\sigma}{100} \right]$$

Next, since for all  $i$  by definition  $\min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \leq 1$ , we get that for all  $j$  as specified in the claim statement, with probability at least 0.8 over the choice of  $T$  and  $T'$  if the verifier didn't reject, it holds that:

$$\mathbb{E}_{i \sim U_{S^j}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \geq \frac{\sigma}{100} + \left( 1 - \frac{\sigma}{100} \right) \left( 1 - \frac{\sigma}{100} \right) \geq 1 - \frac{\sigma}{50} \quad \triangleleft$$

## 72:16 Public Coin Interactive Proofs for Label-Invariant Distribution Properties

▷ Claim 24. Assume the prover's tags satisfy the following inequality:

$$\frac{1}{s} \sum_{i \in [s]: \pi_i \geq \frac{\sigma}{1000N}} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \geq \sigma \quad (24)$$

Then, there exists some  $j_0$  such that  $|S^{j_0}| \geq s \cdot e^{-j_0\tau} \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j_0\tau}}{N} \geq \frac{\varepsilon}{100N}$ , and:

$$\mathbb{E}_{i \sim U_{S^{j_0}}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \leq 1 - 0.7\sigma \quad (25)$$

Proof. We decompose the sum in Inequality (5) according to alleged buckets as follows:

$$\begin{aligned} \sigma &\leq \frac{1}{s} \sum_{i \in [s]: \pi_i \geq \frac{\sigma}{1000N}} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \\ &= \frac{1}{s} \sum_{j: |S^j| \neq \emptyset} \sum_{i \in S^j} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \\ &= \sum_{j: |S^j| \neq \emptyset} \frac{|S^j|}{s} \cdot \frac{1}{|S^j|} \sum_{i \in S^j} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \\ &= \sum_{j: |S^j| \neq \emptyset} \frac{|S^j|}{s} \cdot \mathbb{E}_{i \sim U_{S^j}} \left[ 1 - \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \end{aligned}$$

Define  $J = \left\{ j : |S^j| \geq e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}, \frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N} \right\}$ , and denote  $\sum_{j \notin J} \frac{|S^j|}{s} = \alpha$ . Define next  $J^c = \left\{ j : 0 < |S^j| < e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}, \frac{e^{j\tau}}{N} \geq \frac{\sigma}{1000N} \right\}$ . Observe that:

$$\sum_{j \in J^c} \frac{|S^j|}{s} \leq \frac{1}{s} \sum_{j \in J^c} e^{-j\tau} \cdot s \cdot \frac{\varepsilon \cdot \tau}{100 \log N} \leq \sum_{j \in J^c} \frac{100}{\sigma} \cdot \frac{\varepsilon \cdot \tau}{100 \log N} \leq \frac{\sigma}{20}$$

Then:

$$\sum_{j \in J} \frac{|S^j|}{s} \cdot \mathbb{E}_{i \sim U_{S^j}} \left[ 1 - \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \geq 0.7\sigma$$

Consider thus the distribution  $\mathcal{B}$  that assigns to every  $j \in J$  the probability  $\left| \frac{|S^j|}{s \cdot (1-\alpha)} \right|$ , and 0 otherwise. Then:

$$\mathbb{E}_{j \sim \mathcal{B}} \left[ \mathbb{E}_{i \sim U_{S^j}} \left[ 1 - \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \right] \geq \sigma - \frac{\sigma}{20} \geq 0.9\sigma$$

And so it must hold that there exists some  $j_0 \in J$  such that:

$$\mathbb{E}_{i \sim U_{S^{j_0}}} \left[ 1 - \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \geq 0.9\sigma$$

Finally, this implies that for  $j_0$ :

$$\mathbb{E}_{i \sim U_{S^{j_0}}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \leq 1 - 0.9\sigma \quad \triangleleft$$

▷ **Claim 25.** With high probability over the choice of  $S, T, T'$ , if Inequality (5) holds, then, with high probability, the verifier rejects.

*Proof.* Assume the prover's response  $(\pi_i)_{i \in [s]}$  satisfies Inequality (24). Then, by Claim 24, it holds that there exists some  $j_0$  such that  $|S^{j_0}| \geq s \cdot \frac{\varepsilon \cdot \tau}{100 \log N}$  and  $\frac{e^{j_0 \tau}}{N} \geq \frac{\varepsilon}{100N}$ , and for which:

$$\mathbb{E}_{i \sim U_{S^{j_0}}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \leq 1 - 0.9\sigma \quad (26)$$

Next, by Claim 18, with probability at least 0.98 over the choice of  $T, T'$ , Inequalities (10) and (13) hold for  $j_0$ . Then, assuming the verifier didn't reject, by Claim 23 it holds that with probability at least 0.8 over the choice of  $T, T'$  that:

$$\mathbb{E}_{i \sim U_{S^{j_0}}} \left[ \min \left\{ \frac{D(S_i)}{\pi_i}, \frac{\pi_i}{D(S_i)} \right\} \right] \geq 1 - \frac{\sigma}{50} \quad (27)$$

Note that Inequality (26) and Inequality (27) contradict one another, from which we conclude that if the prover's response satisfies Inequality (24), then with probability at least 0.75 over the choice of  $S, T, T'$ , the verifier should reject. ◁

Finally, concerning the final verifier test:

▷ **Claim 26.** If the prover's answer didn't result with the verifier rejecting the test in Step 4 of Protocol 1, then with probability at most 0.01, Inequality (6) holds.

*Proof.* By Proposition 30  $\mathbb{E}[\tilde{C}_{-\infty}] = s \sum_{i: \pi_i < \frac{\sigma}{1000N}} D(x)$ . Thus, assuming that Inequality (6) holds, every entry in  $T$  has probability at least  $\sum_{i \in [s]: \pi_i < \frac{\sigma}{1000N}} D(x) \geq s \cdot \frac{\sigma}{10N}$  of landing on  $S^{-\infty}$ , and by Hoeffdings Inequality, this will yield:

$$\tilde{C}_{-\infty} \in \left(1 + \frac{1}{\sqrt{s}}\right) s^2 \cdot \frac{\sigma}{10N} > s \cdot |S^-| \cdot \frac{\sigma}{50N}$$

And the verifier rejects with high probability. ◁

### 4.3 From verified uniform tagged sample to property verification

► **Lemma 27.** For every two distributions  $D, Q$  over domain  $[N]$ , and parameter  $\sigma \in (0, 1)$ . Let  $(z_i)_{i \in [s]}$  be a sample of size  $s = \tilde{O}(N^{2/3}) \text{poly}(\sigma^{-1})$  drawn uniformly from  $[N]$ . There exists an algorithm that runs in time  $O(s)$  and outputs  $\delta \in [0, 1]$ , such that  $|\delta - \Delta_{SD}(Q, D)| = O\left(\sigma + \frac{1}{\sqrt{s}}\right)$ , given the following input:

- The sample  $(z_i)_{i \in [s]}$ .
- $(\pi_i)_i \in [0, 1]^s$ , that satisfy the following two inequalities:

$$\frac{1}{s} \sum_{i \in [s]: \pi_i \geq \frac{\sigma}{1000N}} \left( 1 - \min \left\{ \frac{\pi_i}{D(z_i)}, \frac{D(z_i)}{\pi_i} \right\} \right) \leq \sigma \quad (28)$$

$$\frac{1}{s} \sum_{i \in [s]: \pi_i \leq \frac{\sigma}{1000N}} D(z_i) \leq \frac{\sigma}{10N} \quad (29)$$

- $Q(z_i)$ , for all  $i \in [s]$ .

## 72:18 Public Coin Interactive Proofs for Label-Invariant Distribution Properties

**Proof.** Consider the following algorithm: for every  $i \in [S]$ , set  $\theta'_{z_i} = \frac{|\pi_i - Q(z_i)|}{2}$ , and output  $\delta = \frac{1}{s} \sum_{i \in [s]} \theta'_{z_i}$ . We show that this algorithm satisfies the conditions of the lemma.

For every  $x \in [N]$  define  $\theta_x = \frac{|D(x) - Q(x)|}{2}$ . Observe that by definition,  $\Delta_{\text{SD}}(D, Q) = \mathbb{E}_{x \sim U_{[N]}}[\theta_x]$ . Since the sample  $(z_i)$  was drawn i.i.d., the collection  $(\theta_x)$  is independent. By Hoeffding's Inequality:

$$\Pr \left( \left| \frac{1}{s} \sum_{i \in [s]} \theta_{z_i} - \Delta_{\text{SD}}(D, Q) \right| > \frac{2}{\sqrt{s}} \right) \leq 2e^{-8} < 0.01$$

And so, with probability at least 0.99 over the choice of  $(z_i)$ :

$$\left| \frac{1}{s} \sum_{i \in [s]} \theta_{z_i} - \Delta_{\text{SD}}(D, Q) \right| \leq \frac{2}{\sqrt{s}} \quad (30)$$

By assumption over  $(\pi_i)$  and the Triangle Inequality:

$$\left| \frac{1}{s} \sum_{i \in [s]} \theta_{z_i} - \frac{1}{s} \sum_{i \in [s]} \theta'_{z_i} \right| \leq \frac{1}{s} \left| \sum_{i \in [s]} \left( \frac{|D(z_i) - Q(z_i)|}{2} - \frac{|\pi_i - Q(z_i)|}{2} \right) \right| \quad (31)$$

$$\leq \frac{1}{2s} \sum_{i \in [s]} (|D(z_i) - Q(z_i)| - |\pi_i - Q(z_i)|) \quad (32)$$

$$\leq \frac{1}{2s} \sum_{i \in [s]} |(D(z_i) - Q(z_i)) - (\pi_i - Q(z_i))| \quad (33)$$

$$= \frac{1}{2s} \sum_{i \in [s]} |D(z_i) - \pi_i| \quad (34)$$

For every  $i$  such that  $D(z_i) \neq 0$ , it holds that save for at most  $\sigma$ -fraction of  $i \in [s]$ ,  $\pi_i \in (1 \pm O(\sigma)) D(z_i)$ , and for every  $i$  such that  $D(z_i) \neq 0$ , it must hold that  $\frac{1}{s} \sum_{i \in [s]: D(z_i)=0} \pi_i \leq \sigma$ . And so:

$$\frac{1}{2s} \sum_{i \in [s]} |D(z_i) - \pi_i| \leq \frac{1}{2s} \sum_{i \in [s]: D(z_i) \neq 0} D(z_i) \left| 1 - \frac{\pi_i}{D(z_i)} \right| + \frac{1}{2s} \sum_{i \in [s]: D(z_i)=0} \pi_i \quad (35)$$

$$\leq \frac{1}{2} \left( \frac{1}{s} \sum_{i \in [s]: D(z_i) \stackrel{O(\sigma)}{\approx} \pi_i} D(z_i) \left| 1 - \frac{\pi_i}{D(z_i)} \right| + \frac{1}{s} \sum_{i \in [s]: D(z_i) \stackrel{O(\sigma)}{\not\approx} \pi_i} D(z_i) \left| 1 - \frac{\pi_i}{D(z_i)} \right| + \sigma \right) \quad (36)$$

$$\leq \frac{1}{2} (O(\sigma) + O(\sigma) + \sigma) \quad (37)$$

$$= O(\sigma) \quad (38)$$

We thus conclude that with high probability over  $(z_i)$ , the algorithm yields  $\delta$  such that:  $|\delta - \Delta_{\text{SD}}(D, Q)| = O(\sigma + \frac{1}{\sqrt{s}})$   $\blacktriangleleft$

An immediate corollary of this lemma is Theorem 3. We note here that this method can also be leveraged to achieve an efficient protocol for identity testing from an approximate tagged sample *drawn according to*  $D$ , and so, can be also implemented on the output of [14] without incurring further overhead.

We now address the question of verification of label-invariant distribution problems. First, we recall the following definition:

► **Definition 28** (Efficient approximate decision procedure, [13]). *A distribution property  $\mathcal{P}$  has a  $\mu$ -efficient approximate decision procedure if there exists a polynomial-time procedure  $A$  as follows.  $A$  gets as input the domain size  $N$ , a distance parameter  $\sigma \in (0, 1)$ , and a histogram  $(m_j)_j$  satisfying  $\sum_j |m_j - Q(B_j^Q)| \leq \mu$ . For every integer  $N$ , every distribution  $D$  over  $[N]$  and every  $\sigma > 0$ :*

- *If  $Q$  is in  $\mathcal{P}$ , then  $A$  accepts the  $(m_j)_j$ .*
- *$A$  rejects every  $(m_j)_j$  histogram that is consistent with a distribution that is not  $\sigma$ -close to  $\mathcal{P}$ .*

► **Corollary 29.** *Let  $\mathcal{P}$  be a label-invariant distribution property,  $0 \leq \varepsilon_c < \varepsilon_f \leq 1$  distance parameters, and assume  $\mathcal{P}$  admits an efficient  $\tau$ -approximate decision procedure, where  $\tau = O(\varepsilon_f - \varepsilon_c)^3$ . Given sample access to distribution  $D$  over domain  $[N]$ , there exists a 2-message public-coin protocol with verifier sample complexity and communication complexity  $\tilde{O}(N^{2/3}) \cdot \text{poly}(\tau^{-1})$ , such that:*

- **Completeness.** *If  $\Delta_{SD}(D, \mathcal{P}) \leq \varepsilon_c$ , the verifier accepts with high probability.*
- **Soundness.** *If  $\Delta_{SD}(D, \mathcal{P}) \geq \varepsilon_f$ , the verifier rejects with high probability.*

We outline how to obtain a protocol for every label-invariant distribution property admitting an efficient decision procedure from a uniform verified tagged sample. Generally, we follow [13]. The reader is referred to their work for further detail on efficient decision procedures, as well as examples for such procedures for natural label-invariant properties, such as those relating to Shannon entropy, support size, and distance from uniformity. We note that the main obstacle in the protocol behind the above corollary, addressed by this paper in a novel way, is obtaining a good approximation of the probability according to  $D$  of randomly chosen elements in the domain. Recall that without communication, this task requires  $\tilde{O}(N)$  samples and runtime from the verifier.

We provide an outline the protocol behind Corollary 29. The verifier and the prover run Protocol 1 over distribution  $D$  with distance parameter  $\sigma = \frac{\varepsilon_f - \varepsilon_c}{3}$ , and with the following addition: the prover also sends, alongside  $(\pi_i)_{i \in [s]}$ , the tags  $(q_i)_{i \in [s]}$ , such that for all  $i \in [s]$ ,  $q_i = Q(i)$ , for some distribution  $Q \in \mathcal{P}$ . The verifier performs the following checks:

1. The verifier runs the tests outlined in Protocol 1 with respect to  $(\pi_i)$ , and rejects w.h.p. if prover tags satisfy Inequalities (5) or (6).
2. The verifier uses  $(q_i)$  to compute the bucket histogram of distribution  $Q$ . The size of every bucket of significant mass  $j$  of  $Q$  can be approximated to high accuracy from a uniform tagged sample  $(q_i)$ . Then, the mass of each bucket can be approximated by the product of the size and  $\frac{e^{j\tau}}{N}$ . Note that this process yields a probability histogram for  $Q$  that is accurate with high probability up to  $\tau$  multiplicative factor. Then, the verifier runs the  $\tau$ -approximate decision procedure with distance parameter  $\sigma$ , to check that indeed  $Q \in \mathcal{P}_N$ , and reject if it's far.
3. If non of the above tests failed, the verifier estimates the distance between  $Q$  and  $D$  using  $(\pi_i)$  and  $(q_i)$  as outlined in Lemma 27, and rejects unless estimate smaller than  $\varepsilon_c + O(\tau)$ .

If the all tests passed, then with high probability it holds that  $Q$  is  $\tau$ -close to  $\mathcal{P}$ , and that  $\Delta_{SD}(Q, D) \leq \varepsilon_c + O(\tau)$ , and the conditions of Corollary 29 hold. If  $D$  is  $\varepsilon_f$  far from the property, and the tags  $(q_i)$  produce a histogram consistent with a histogram of a distribution that passes the efficient decision procedure, then by assumption, it holds that there exists some  $Q \in \mathcal{P}$  that is  $\varepsilon_f - \tau$  far from  $D$ , and so the distance test will fail. We omit further detail, as the process of verifying membership in distribution property from approximate histogram is outlined in [13].

## References

- 1 Gal Arnon and Guy N. Rothblum. On prover-efficient public-coin emulation of interactive proofs. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPICs*, pages 3:1–3:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ITC.2021.3.
- 2 László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988. doi:10.1016/0022-0000(88)90028-1.
- 3 Tugkan Batu and Clément L. Canonne. Generalized uniformity testing. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 880–889. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.86.
- 4 Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988. doi:10.1007/0-387-34799-2\_4.
- 5 Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. 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 53:1–53:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.ITCS.2018.53.
- 6 Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004. doi:10.1016/j.ic.2003.09.005.
- 7 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. doi:10.1007/3-540-47721-7\_12.
- 8 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.
- 9 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991. doi:10.1145/116825.116852.
- 10 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985. doi:10.1145/22145.22178.
- 11 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68. ACM, 1986. doi:10.1145/12130.12137.
- 12 Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. *Comput. Complex.*, 27(1):99–207, 2018. doi:10.1007/s00037-016-0136-9.
- 13 Tal Herman and Guy N. Rothblum. Verifying the unseen: interactive proofs for label-invariant distribution properties. 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 1208–1219. ACM, 2022. doi:10.1145/3519935.3519987.
- 14 Tal Herman and Guy N. Rothblum. Doubly-efficient interactive proofs for distribution properties. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 743–751. IEEE, 2023. doi:10.1109/FOCS57990.2023.00049.

- 15 Tal Herman and Guy N. Rothblum. Interactive proofs for general distribution properties. *Electron. Colloquium Comput. Complex.*, pages TR24–094, 2024. URL: <https://eccc.weizmann.ac.il/report/2024/094>.
- 16 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.
- 17 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam D. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM J. Comput.*, 39(3):813–842, 2009. doi:10.1137/070701649.
- 18 Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802. ACM, 2013. doi:10.1145/2488608.2488709.
- 19 Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 200–207. ACM, 2000. doi:10.1145/335305.335330.
- 20 Gregory Valiant and Paul Valiant. The power of linear estimators. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 403–412. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.81.

## A Collision Tests Analysis

### A.1 Twoway Collisions

► **Proposition 30.** *Assume that for every  $x \in [N]$ ,  $D(x) \leq \frac{1}{s}$ . For every sample  $S$  such that for every  $i \in [s]$ , the element  $S_i$  appears at most  $\log N$  times in  $S$ , with probability at least  $1 - \frac{\tau}{100 \log N}$  over the choice of the sample  $T$ , it holds that:*

$$\mathbb{E} \left[ \tilde{C}_j^{pair} \right] = s \sum_{i \in S^j} D(S_i)$$

As well as:

$$\left| \tilde{C}_j^{pair} - \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{pair} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{pair} \right]}}$$

**Proof.** The reader is referred to the Appendix of Herman and Rothblum [14] for a detailed proof of this claim. In a nutshell, For every  $k, r \in [s]$  denote by  $C_{k,r}$  the indicator of the event  $\{S_k = T_r\}$ . Observe that  $\tilde{C}_j^{pair} = \sum_{k \in S^j} \sum_{r \in [s]} C_{k,r}$ , and that  $\mathbb{E}_T [C_{k,r}] = D(S_k)$ . By the linearity of expectation:

$$\mathbb{E} \left[ \tilde{C}_j^{pair} \right] = \sum_{k \in S^j} \sum_{r \in [s]} \mathbb{E} [C_{k,r}] = \sum_{k \in S^j} \sum_{r, r' \in [s]} D(S_k) = s \sum_{k \in S^j} D(S_k) \quad (39)$$

In order to prove concentration we show that  $\text{Var}_T \left[ \tilde{C}_j^{pair} \right]$  is small. Herman and Rothblum [14] show that the variance can be bounded by  $\log N \mathbb{E} \left[ \tilde{C}_j \right]$ . And so, the desired result is thus achieved through Chebychevs' Inequality. ◀

## A.2 Threeway Collisions

► **Proposition 31.** *Assume that for every  $x \in [N]$ ,  $D(x) \leq \frac{1}{s}$ . For every sample  $S = (S_i)_{i \in [s]}$  such that for every  $i \in [s]$ , the element  $S_i$  appears in at most  $\log N$  locations in  $S$ , with probability at least  $1 - \frac{\tau}{100 \log N}$  over the choice of the samples  $T, T'$ , it holds that for any set of bucket indices  $J$  of size at most  $2 \frac{\log N}{\tau}$ , for every  $j \in J$ :*

$$\mathbb{E} \left[ \tilde{C}_j^{triple} \right] = s^2 \sum_{i \in S^j} (D(x))^2$$

As well as:

$$\left| \tilde{C}_j^{triple} - \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \right| \leq \mathbb{E} \left[ \tilde{C}_j^{triple} \right] \cdot \sqrt{\frac{300 \log^2 N}{\tau \cdot \mathbb{E} \left[ \tilde{C}_j^{triple} \right]}}$$

**Proof.** For every  $k, r, r' \in [s]$  denote by  $C_{k,r,r'}$  the indicator of the event  $\{S_k = T_r = T'_{r'}\}$ . Observe that  $\tilde{C}_j^{triple} = \sum_{k \in S^j} \sum_{r, r' \in [s]} C_{k,r,r'}$ , and that  $\mathbb{E}_{T, T'} [C_{k,r,r'}] = (D(S_k))^2$ . By the linearity of expectation:

$$\mathbb{E} \left[ \tilde{C}_j^{triple} \right] = \sum_{k \in S^j} \sum_{r, r' \in [s]} \mathbb{E}_{T, T'} [C_{k,r,r'}] = \sum_{k \in S^j} \sum_{r, r' \in [s]} (D(S_k))^2 = s^2 \sum_{k \in S^j} (D(S_k))^2 \quad (40)$$

Next, we show that for every  $j \in J$  the random variable  $\tilde{C}_j^{triple}$  is well concentrated around its mean. In order to do so, we bound the variance of  $\tilde{C}_j^{triple}$ . Note that:

$$\text{Var} \left[ \tilde{C}_j^{triple} \right] = \sum_{\substack{(k_0, r_0, r'_0) \in [s]^3 \\ (k_1, r_1, r'_1) \in [s]^3}} \text{Cov} [C_{k_0, r_0, r'_0}, C_{k_1, r_1, r'_1}]$$

And so, in order to bound the variance, consider the following case analysis for the pair  $((k_0, r_0, r'_0), (k_1, r_1, r'_1))$ :

- **Type I.**  $S_{k_0} \neq S_{k_1}$ , then: either  $r_0 \neq r_1$  and  $r'_0 \neq r'_1$  in which case  $\text{Cov} [C_{k_0, r_0, r'_0}, C_{k_1, r_1, r'_1}] = 0$  as the variables are independent; or  $r_0 = r_1$  or  $r'_0 = r'_1$ , in which case since  $S_{k_0} \neq S_{k_1}$ , it cannot be that  $C_{k_0, r_0, r'_0} = 1$  and  $C_{k_1, r_1, r'_1} = 1$  simultaneously, which means that  $\text{Cov} [C_{k_0, r_0, r'_0}, C_{k_1, r_1, r'_1}] < 0$ .
- **Type II.**  $S_{k_0} = S_{k_1}$  and  $(r_0, r'_0) = (r_1, r'_1)$ , then  $\text{Cov} [C_{k_0, r_0, r'_0}, C_{k_1, r_1, r'_1}] = \text{Var} [C_{k_0, r_0, r'_0}] \leq \mathbb{E} [C_{k_0, r_0, r'_0}]$ .

■ **Type III.:**

- **Type IIIa.**  $S_{k_0} = S_{k_1}$  and  $r_0 = r_1 = r$ , however  $r'_0 \neq r'_1$ , then:

$$\text{Cov} [C_{k_0, r, r'_0}, C_{k_1, r, r'_1}] \leq \mathbb{E} [C_{k_0, r, r'_0} \cdot C_{k_1, r, r'_1}] = (D(S_{k_0}))^3$$

- **Type IIIb.**  $S_{k_0} = S_{k_1}$  and  $r'_0 = r'_1 = r'$ , however  $r_0 \neq r_1$ , then:

$$\text{Cov} [C_{k_0, r, r'_0}, C_{k_1, r, r'_1}] \leq \mathbb{E} [C_{k_0, r_0, r'} \cdot C_{k_1, r_1, r'}] = (D(S_{k_0}))^3$$

Since all pairs of indicators of Type I do not contribute to the variance, we are left to quantify how many pairs of indicators are there of Type II and Type III. Fix  $k_0 \in [s]$ , and denote  $A_{k_0} = \{i \in [s] : S_i = S_{k_0}\}$ .

- **Type II.** By assumption over  $S$ ,  $|A_{k_0}| \leq \log N$ , and so, there are at most  $\log N$  options for  $k_1$ . Then, there are  $s^2$  ways to pick  $(r, r')$ . Therefore,  $k_0$  participates in at most  $s^2 \cdot \log N$  pairs of Type II.



- **Type III.** This type is divided into two symmetric sub-types. As above, for a fixed  $k_0$ , there are at most  $\log N$  possible values for  $k_1$ . Then, there are  $s^3$  ways to pick  $r, r'_0, r'_1$ . Therefore,  $k_0$  participates in at most  $2 \cdot s^3 \cdot \log N$  pairs of Type IIIa. Type IIIb is the symmetric where both triplets agree on  $r'$ , but have two different values  $r_0$  and  $r_1$ .

First, we calculate the contribution of all the *Type II* pairs to the variance:

$$\sum_{(k_0, r, r') \in [s]^3} \sum_{k_1 \in A_{k_0}} \text{Cov} [C_{k_0, r, r'}, C_{k_1, r, r'}] \leq \sum_{(k_0, r, r') \in [s]^3} \sum_{k_1 \in A_{k_0}} \mathbb{E} [C_{k_0, r, r'}] \quad (41)$$

$$\leq \log N \sum_{(k_0, r, r') \in [s]^3} \mathbb{E} [C_{k_0, r, r'}] \quad (42)$$

$$= \log N \cdot \mathbb{E} [\tilde{C}_j^{triple}] \quad (43)$$

As for the *Type IIIa* pairs:

$$\sum_{(k_0, r, r'_0, r'_1) \in [s]^4} \sum_{k_1 \in A_{k_0}} \text{Cov} [C_{k_0, r, r'_0}, C_{k_1, r, r'_1}] \leq \sum_{(k_0, r, r'_0, r'_1) \in [s]^4} \sum_{k_1 \in A_{k_0}} (D(S_{k_0}))^3 \quad (44)$$

$$\leq \log N \sum_{(k_0, r, r'_0, r'_1) \in [s]^4} (D(S_{k_0}))^3 \quad (45)$$

$$\leq s \cdot \log N \sum_{(k_0, r, r'_0, r'_1) \in [s]^3} (D(S_{k_0}))^3 \quad (46)$$

$$= \log N \cdot \mathbb{E} [\tilde{C}_j^{triple}] \quad (47)$$

Similarly, all *Type IIIb* contribute at most  $\log N \cdot \mathbb{E} [\tilde{C}_j^{triple}]$  to the variance as well. We thus conclude that:

$$\text{Var} [\tilde{C}_j^{triple}] \leq 3 \log N \cdot \mathbb{E} [\tilde{C}_j^{triple}]$$

Therefore, using Chebichev's Inequality:

$$\Pr_{T, T'} \left( \left| \tilde{C}_j^{triple} - \mathbb{E} [\tilde{C}_j^{triple}] \right| \geq \sqrt{\frac{300 \log^2 N}{\tau} \cdot \mathbb{E} [\tilde{C}_j^{triple}]} \right) \leq \frac{3 \log N \cdot \mathbb{E} [\tilde{C}_j^{triple}]}{\frac{300 \log^2 N}{\tau} \cdot \mathbb{E} [\tilde{C}_j^{triple}]} \quad (48)$$

$$\leq \frac{\tau}{100 \log N} \quad (49)$$

Taking union bound over all  $j \in J$  yields the desired result.  $\blacktriangleleft$



# Additive Noise Mechanisms for Making Randomized Approximation Algorithms Differentially Private

Jakub Tětek     
INSAIT, Sofia, Bulgaria

---

## Abstract

---

The exponential increase in the amount of available data makes taking advantage of them without violating users' privacy one of the fundamental problems of computer science. This question has been investigated thoroughly under the framework of differential privacy. However, most of the literature has not focused on settings where the amount of data is so large that we are not even able to compute the exact answer in the non-private setting (such as in the streaming setting, sublinear-time setting, etc.). This can often make the use of differential privacy unfeasible in practice.

In this paper, we show a general approach for making Monte-Carlo randomized approximation algorithms differentially private. We only need to assume the error  $R$  of the approximation algorithm is sufficiently concentrated around 0 (e.g.  $\mathbb{E}[|R|]$  is bounded) and that the function being approximated has a small global sensitivity  $\Delta$ . Specifically, if we have a randomized approximation algorithm with sufficiently concentrated error which has time/space/query complexity  $T(n, \rho)$  with  $\rho$  being an accuracy parameter, we can generally speaking get an algorithm with the same accuracy and complexity  $T(n, \Theta(\epsilon\rho))$  that is  $\epsilon$ -differentially private.

Our technical results are as follows. First, we show that if the error is subexponential, then the Laplace mechanism with error magnitude proportional to the sum of the global sensitivity  $\Delta$  and the *subexponential diameter* of the error of the algorithm makes the algorithm differentially private. This is true even if the worst-case global sensitivity of the algorithm is large or infinite. We then introduce a new additive noise mechanism, which we call the zero-symmetric Pareto mechanism. We show that using this mechanism, we can make an algorithm differentially private even if we only assume a bound on the first absolute moment of the error  $\mathbb{E}[|R|]$ .

Finally, we use our results to give either the first known or improved sublinear-complexity differentially private algorithms for various problems. This includes results for frequency moments, estimating the average degree of a graph in sublinear time, rank queries, or estimating the size of the maximum matching. Our results raise many new questions and we state multiple open problems.

**2012 ACM Subject Classification** Security and privacy; Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** Differential privacy, Randomized approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.73

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2211.03695>

**Funding** This work was done while the author was at BARC, University of Copenhagen. He was supported by the VILLUM Foundation grant 54451. This research was partially funded from the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

**Acknowledgements** The author would like to thank Rasmus Pagh and anonymous reviewers for helping to improve this paper.



© Jakub Tětek;

licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 73; pp. 73:1–73:20



Leibniz International Proceedings in Informatics

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

## 1 Introduction

With the increase in the amount of available data, the problem of analyzing it in a privacy-preserving manner has become a central problem in computer science. One commonly used tool for this task is differential privacy, which is a well-established notion of privacy that is commonly used in data analysis and machine learning. However, with some notable exceptions, the literature on differential privacy has focused on the setting where the amount of data is small enough that we would be able to practically solve a given problem exactly in the non-private setting. However, in practice, this assumption is often not realistic – this is after all the reason for the existence of (among others) streaming and sublinear-time algorithms.

Our goal is thus to get very efficient (sublinear) algorithms that at the same time guarantee differential privacy. In the streaming or sublinear-time setting, the error coming from the algorithm not being exact will generally speaking be much bigger than the amount of noise that the given problem necessitates for ensuring differential privacy. The main objective in this setting is thus not to simply minimize the amount of noise we add, but rather to achieve a given level of accuracy while minimizing the complexity (e.g. space, time, or query complexity) of the algorithm. Of course, some amount of noise inherently has to be added to achieve privacy, but this is usually so small, that one would need linear complexity to get such a level of accuracy even without privacy. In the sublinear regime, we thus usually do not have to worry whether a given level of accuracy is achievable and we instead focus as our central objective on the complexity needed to achieve it.

One of the main difficulties in making sublinear algorithms private is that most sublinear-time and streaming algorithms are randomized and give only probabilistic guarantees on the quality of the output. This makes adding noise based on global sensitivity<sup>1</sup> – which is commonly used to get differentially private algorithms – unsuitable for this situation, as in the worst case the global sensitivity of the approximation algorithm<sup>2</sup> can be very large even if the global sensitivity of the function being approximated is small. In this paper, we propose a way to get around this issue by showing additive noise mechanisms that only need that (1) the function being approximated has low global sensitivity and (2) the answer of the algorithm is sufficiently concentrated around the true value.

We give two main results. The first one states that if the error has subexponential tails<sup>3</sup>, adding the Laplace distribution suffices to achieve pure differential privacy – this can be seen as a generalization of the standard Laplace mechanism. The second result shows a similar result for a different distribution, and it only assumes bounded mean deviation (or higher moments) of the error instead of being subexponential. While the first result has much stronger assumption about the error distribution, it is also stronger in that it also works for multiple adaptive queries that are *not answered independently*. This is useful for example for streaming algorithms, where multiple queries can be answered using a single sketch and are thus not answered independently. Note that the standard composition theorem would allow us to perform multiple queries only if they were answered independently.

We use our results to give new differentially private algorithms for various problems: for maximum matching under node-level privacy, frequency moments, counting connected components under edge-level privacy, and rank queries. We also show how a common

---

<sup>1</sup> Global sensitivity of a function  $g$  with respect to a relation  $\sim$  is defined as  $\sup_{x \sim x'} |g(x) - g(x')|$ .

<sup>2</sup> Here, we see the approximation algorithm as a deterministic function of the input and a string of random bits.

<sup>3</sup> A distribution is subexponential if its tails are dominated by the tail of an exponential distribution.

technique for designing relative-approximation sublinear-time algorithms – *advice removal by geometric search* – can be made differentially private. This implies an edge-differentially-private algorithm for estimating the average degree of a graph, improving upon the state of the art [5], but we think it could also be useful for many other problems. Our algorithm for maximum matching also answers an open problem from [5]. For the other mentioned problems, we give the most efficient differentially private algorithm known.

Notably, a concurrent work [7] gave a similar result. In the context of global sensitivity, their result makes weaker assumptions and results in algorithms that are less efficient since they rely on postprocessing to turn approximate differential privacy into pure differential privacy. We discuss this in detail in Section 1.3.

## 1.1 Our results

We now informally state our main technical results and then we state the results for specific results that we obtained using the technical results. We also show how to use our technical results to give algorithms for specific problems in Section 1.2. We start with a result for algorithms with subexponential error tails, which appears in Section 3:

► **Theorem 9, simplified version.** Assume we have an algorithm  $A(D)$  for  $D$  being a dataset. Assume there exists a function  $g$  with global sensitivity  $\leq \Delta_1$  w.r.t.  $D$  such that  $A(D) - g(D)$  has *subexponential diameter*<sup>4</sup>  $\leq \Delta_2$ , i.e.  $\mathbb{P}[|A(D) - g(D)| \geq t] \leq 2e^{-t/\Delta_2}$  for  $t \geq 0$ . Then releasing  $A(D) + \text{Laplace}(O((\Delta_1 + \Delta_2)/\epsilon))$  is  $\epsilon$ -differentially private for  $\epsilon \leq O(1)$ .

Moreover, with noise  $\text{Laplace}(O(k(\Delta_1 + \Delta_2)/\epsilon))$ , this also holds if we make  $k$  such releases with different algorithms  $A_1, \dots, A_k$  chosen adaptively that are executed with the same randomness.

Note that this generalizes the claim that the Laplace mechanism with noise magnitude proportional to the global sensitivity gives differential privacy (this can be seen by setting  $A(D) = g(D)$ ). Note also that in the second half, the algorithms use the same randomness, and we thus cannot get this part of the result by standard composition.

We then prove in Section 4 that in the case of a single query, it is sufficient to assume a bound on some deviation moments (multiple independently answered queries can be handled using the standard composition theorem).

► **Theorem 13, simplified version.** *Let us have an algorithm  $A$  such that there exists a function  $g$  with global sensitivity  $\leq \Delta$  and such that  $\mathbb{E}[|A(D) - g(D)|^3] \leq \Delta^3$  for any dataset  $D$ . Then for  $\epsilon \leq O(1)$  there exists a random variable  $Y$  with  $\mathbb{E}[|Y|] \leq O(\Delta/\epsilon)$ , such that  $A(\cdot) + Y$  is  $\epsilon$ -differentially private.*

We now state results which, as we show in Section 5, follow from the above technical results. For all the following results, our algorithm is the most efficient known, except for the streaming algorithm for rank queries, which is incomparable with the algorithm from [19]. With the result for maximum matching, we (together with the concurrent [7]) answer positively the open question of Blocki, Grigorescu, and Mukherjee [5], and the algorithm for estimating the number of edges improves upon the algorithm from [5]. For more detailed comparison with [7], see Section 1.3.

<sup>4</sup> The subexponential diameter of a distribution roughly corresponds to the smallest parameter  $\sigma$ , such that the tail of  $\text{Laplace}(1/\sigma)$  dominates the tails of the distribution.

► **Corollaries 16–20, simplified versions.** *There exists an  $\epsilon$ -differentially private*

1. *streaming algorithm for the frequency  $F_2$  moment problem with space complexity  $O(\frac{1}{\rho^2 \epsilon^2})$  (Corollary 16),*
2. *sublinear-time algorithm for estimating the number of connected components with time complexity  $O(\frac{1}{\rho^2 \epsilon^2} \log \frac{1}{\rho \epsilon})$  (Corollary 17),*
3. *sublinear-time algorithm for estimating the size of the maximum matching with time complexity  $d^{O(1/(\rho^2 \epsilon^2))} / (\rho \epsilon)^{O(1/(\rho \epsilon))}$  (Corollary 18),*
4. *streaming algorithm for answering  $k$  (adaptive) rank queries in space  $O(\frac{k \log^2 k}{\rho \epsilon})$  (Corollary 19),*
5. *sublinear-time algorithm for approximating the number of edges in a graph in time  $O(\frac{n}{\epsilon^2 \rho^2 \sqrt{m}})$  (Corollary 20).*

## 1.2 Our techniques

We now sketch the techniques that we use in our paper. In each part of this section, we also give a reference to the corresponding section of this paper.

### 1.2.1 Subexponential error, one query (Section 3)

Suppose we have a randomized algorithm  $A(D)$  for  $D$  being a dataset that approximates a function  $g(D)$  with global sensitivity  $\leq \Delta_1$  for some parameter  $\Delta_1$ . Define the error  $R$  as the random variable  $R = A(D) - g(D)$  and assume that it is tightly concentrated around 0, namely  $\mathbb{P}[|R| > t] \leq 2e^{-t/\Delta_2}$  for some value  $\Delta_2$  ( $\Delta_2$  is an upper bound on the “subexponential diameter” of  $R$ ). Intuitively speaking,  $\Delta_2$  determines the “scale” of  $R$ , and  $\Delta_2$  is in fact up to a constant factor an upper bound on  $\mathbb{E}[|R|]$ . Note that the tails of  $R$  decrease at least at the same rate as those of the Laplace distribution. This suggests Laplacian noise with large enough magnitude could “hide  $R$ ”. Indeed, we prove that Laplacian noise will guarantee privacy. We now sketch the proof.

**High-level view.** Assume for simplicity that both  $\Delta_1, \Delta_2 \leq 1$ . The same approach works for general values of  $\Delta_1, \Delta_2$  by simple re-scaling. Let  $Y \sim \text{Laplace}(c/\epsilon)$  for appropriately chosen value of  $c$ . We will prove that for any random variable  $X$  with subexponential diameter  $\leq 3$ <sup>5</sup> (that is  $\mathbb{P}[|X| \geq t] \leq 2e^{-t/3}$ ), the probability density functions satisfy  $f_{X+Y}(y)/f_Y(y) = e^{\pm\Theta(\epsilon)}$  for any  $y$ . We can use this to prove privacy, as we now show. For two neighboring datasets  $D_1, D_2$ , we set  $R_1 = A(D_1) - g(D_1)$  and  $R_2 = A(D_2) - g(D_1)$  (note the asymmetry in the definitions). It then holds that  $R_1$  has subexponential diameter  $\Delta_2 \leq 1$ . One can also show that the subexponential diameter of  $R_2 = (A(D_2) - g(D_2)) + (g(D_2) - g(D_1))$  is  $\leq 3$  (Lemma 5). Let  $y' = y - g(D_1)$ . It then holds for any  $y$  that

$$\frac{f_{A(D_1)+Y}(y)}{f_{A(D_2)+Y}(y)} = \frac{f_{g(D_1)+R_1+Y}(y)}{f_{g(D_1)+R_2+Y}(y)} = \frac{f_{R_1+Y}(y')}{f_{R_2+Y}(y')} = \frac{f_{R_1+Y}(y')}{f_Y(y')} \cdot \frac{f_Y(y')}{f_{R_2+Y}(y')} = e^{\pm\Theta(\epsilon)}$$

where the last equality uses the claim  $f_{X+Y}(y')/f_Y(y') = e^{\pm\Theta(\epsilon)}$  for  $X = R_1$  and for  $X = R_2$ . This implies differential privacy.

<sup>5</sup> We choose value 3 as this is the value we will need below. The claim holds also for larger constants with  $c$  chosen appropriately.

**Bounding ratios of density functions.** We now sketch why  $f_{X+Y}(y)/f_Y(y) = e^{\pm\Theta(\epsilon)}$ . Since  $Y$  is continuous, we may re-write

$$f_{X+Y}(y) = \mathbb{E}[f_Y(y - X)] = \frac{\epsilon}{2c} \mathbb{E}[e^{-\epsilon|X-y|/c}] = (*)$$

where the first equality is a standard identity [16]. We now use the inequalities  $|X - y| \leq |X| + |y|$  and  $|X - y| \geq |y| - |X|$ . This allows to bound

$$\begin{aligned} (*) &\leq \frac{\epsilon}{2c} \mathbb{E}[e^{-\epsilon(|y|-|X|)/c}] = \frac{\epsilon}{2c} e^{-\epsilon|y|/c} \mathbb{E}[e^{\epsilon|X|/c}] \\ (*) &\geq \frac{\epsilon}{2c} \mathbb{E}[e^{-\epsilon(|X|+|y|)/c}] = \frac{\epsilon}{2c} e^{-\epsilon|y|/c} \mathbb{E}[e^{-\epsilon|X|/c}] \end{aligned}$$

while it holds that  $f_Y(y) = \frac{\epsilon}{2c} e^{-\epsilon|y|/c}$ . It is thus sufficient to prove that  $\mathbb{E}[e^{\epsilon|X|/c}] \leq e^{O(\epsilon)}$  and  $\mathbb{E}[e^{-\epsilon|X|/c}] \geq e^{-O(\epsilon)}$ . While the first inequality is standard, the second is not. We will now sketch a proof for both.

**Bounding the expectation of exponentials of a subexponential random variable.** If we knew the density function of  $X$ , we could easily express the expectations as integrals. However, not only we do not have a bound on the density, but the density may even not exist. We thus use the following trick. We use the fact that for any real random variable  $Z$ , it holds that  $Z$  has the same distribution as  $F_Z^{-1}(u)$  for  $u \sim \text{Unif}(0, 1)$  where  $F_Z$  is the cumulative distribution function (CDF) of  $Z$ . This allows us to write  $\mathbb{E}[e^{-\epsilon|X|/c}] = \mathbb{E}_u[e^{-F_{\epsilon|X|/c}^{-1}(u)}]$  and similarly for  $\mathbb{E}[e^{\epsilon|X|/c}]$ .

Unlike the density function, we do have a bound on the cumulative distribution function. Specifically, we are assuming  $\mathbb{P}[|X| > t] \leq 2e^{-t/3}$  which implies that  $F_{\epsilon|X|/c}^{-1}(u) \leq -\frac{3\epsilon}{c} \log(\frac{1-u}{2})$ . Upper-bounding the CDF like this reduces the problem to computing the expectation of a function of the uniform random variable, which can be done straightforwardly. This proves the desired bounds.

### 1.2.2 Subexponential error, multiple queries (Section 3)

We would like to be able to release answers to multiple queries which are not answered independently (such as if they are answered based on the same sketch). Since the answers are not independent, we cannot use the composition theorem. We now sketch an alternative approach.

For fixed queries, the above proof goes through with minor modifications even in the multivariate case. Instead of using the inequalities  $|y - X| \leq |y| + |X|$  and  $|y - X| \geq |y| - |X|$  in the case of  $y, X \in \mathbb{R}$ , we use the analogous version for  $\ell_1$  norms in the case of  $y, X \in \mathbb{R}^k$ :  $\|y - X\|_1 \leq \|y\|_1 + \|X\|_1$  and  $\|y - X\|_1 \geq \|y\|_1 - \|X\|_1$ . We then use that if  $X$  is a vector of  $k$  subexponential random variables with diameter  $\leq \Delta$ , then  $\|X\|_1$  has subexponential diameter  $\leq 3k\Delta$  (Lemma 5).

This however gives the result only in the nonadaptive case, when the queries do not depend on the released values: the identity  $f_{X+Y}(y) = \mathbb{E}[f_Y(X - y)]$  relies crucially on  $X = (X_1, \dots, X_k)$  and  $Y = (Y_1, \dots, Y_k)$  being independent. In the case of adaptive queries,  $X_i$  could depend on  $Y_1, \dots, Y_{i-1}$  (the query that we perform – and thus also the answer to it – can be influenced by the noise we add to the previous answers). We instead use our non-adaptive version of the claim and make it adaptive in a black box fashion, by proving a claim that may be of independent interest: If we have a countable number of mechanisms such that releasing the answers of any *fixed* subset of size  $k$  is  $\epsilon$ -differentially private, then we may also pick this subset *adaptively* and it will still be  $\epsilon$ -differentially private.

### 1.2.3 Error with polynomial tails (Section 4)

In the case that the error has polynomial tails, we only consider the case of a single query. Our techniques do not seem to generalize to the multivariate case, and we conjecture that this is impossible (see Section 6). The case when multiple queries are answered independently may be still handled by the standard composition theorem.

An approach similar to the one described above can be made to work, with the difference that we use the inequality  $|x - X| \geq \max(0, |x| - |X|)$  instead of the weaker  $|x - X| \geq |x| - |X|$ . This approach, however, requires proving the following inequality for all  $y, s \geq 0, \alpha > 1, 0 \leq \epsilon \leq 1$ :

$$\int_0^1 \min \left( (1 + |y|/s)^\alpha, \left| 1 - \frac{(1 - 2^{-1/\alpha})\epsilon}{(1 + |y|/s)(1 - u)^{1/\alpha}} \right|^{-\alpha} \right) du \leq 1 + \frac{2\alpha - 1}{\alpha - 1} \epsilon.$$

This is the technically most challenging part of this paper. The trick is to bound the inside of the integral for  $u \in [0, 1 - \epsilon(1 + |y|/s)^{-\alpha}]$  by a simpler expression that can be successfully integrated. The rest of the interval  $[0, 1]$  contributes at most  $\epsilon$ , as its length is  $\epsilon(1 + |y|/s)^{-\alpha}$  and the maximum value of the function being integrated is  $\leq (1 + |y|/s)^\alpha$ .

### 1.2.4 How to use our technical results (Section 5)

#### 1.2.4.1 Error with subexponential tails

We now sketch how one can use Theorem 9 to get differentially private mechanisms for specific problems. Recall that the theorem states that if our algorithm approximates a function with global sensitivity  $\leq \Delta_1$  and the subexponential diameter of the error is  $\leq \Delta_2$ , then adding noise from  $Laplace(O(\Delta_1 + \Delta_2)/\epsilon)$  makes the algorithm  $\epsilon$ -differentially private.

We start with a randomized approximation algorithm whose error is subexponentially concentrated around zero (often, this is either known or easy to prove) that approximates a parameter with a small global sensitivity  $\Delta_1$ . This is the case for example for the YYI maximum matching algorithm [34] under node-level privacy or the KLL sketch [20] for rank queries. Suppose the complexity (such as time/space/query/sample or other complexity) of the algorithm is  $T(n, \rho)$  and has additive error of scale (i.e. with subexponential diameter)  $\rho n$ . If we want the final error with privacy to be  $O(\rho n)$ , then we run the algorithm with error parameter  $\epsilon \rho$ , making the error's subexponential diameter be  $O(\epsilon \rho n)$ . We then get from Theorem 9 that adding noise of magnitude  $O(\rho n)$  is sufficient to get  $\epsilon$ -differential privacy, assuming  $\Delta_1$  is sufficiently small, giving us the desired result. The complexity of the private algorithm will thus be  $T(n, \Theta(\epsilon \rho))$ . This allows us to achieve a given level of accuracy<sup>6</sup> under pure differential privacy, while not significantly worsening the algorithm's complexity. We describe this approach in greater detail and with general failure probabilities (not just constant) in Section 5.

To illustrate the second part of the theorem, consider for example a rank queries sketch (see Section 5.5 for details). The algorithms  $A_1, \dots, A_k$  correspond to making  $k$  adaptive queries to the sketch of a dataset (assume we are given  $k$  queries we need to perform) and the fact that the algorithms use the same randomness corresponds to us querying the same sketch (as compared to  $k$  independent sketches). Specifically, the algorithm  $A_i$  here builds

<sup>6</sup> This is true unless  $\rho$  is very small, as otherwise the global sensitivity will necessitate some level of noise. As we noted, this usually happens only when  $T(n, \epsilon \rho) \geq \Omega(n)$ , making this uninteresting for our sublinear setting.



the sketch (using the shared randomness), and then performs the  $i$ -th query. Note that the fact that we only use one sketch *prevents us from using the composition theorem* to get this from just the first part of the theorem.

### 1.2.4.2 Error with polynomial tails

If the error has polynomial tails, we can make it private in a way essentially the same as in the subexponential case using Theorem 13. Moreover, it holds that if we have  $\mathbb{E}[|A(D) - g(D)|] \leq \Delta$ , then by taking a median of 5 independent executions of  $A$ , we get an algorithm  $A'$  whose error's third moment is also bounded:  $\mathbb{E}[|A'(D) - g(D)|^3] \leq O(\Delta^3)$  [21]. This means that we can make an algorithm private even if we only have a bound on  $\mathbb{E}[|A(D) - g(D)|]$ , for  $g$  being a low-global-sensitivity function, or the mean squared error  $\mathbb{E}[(A(D) - g(D))^2]$ .

## 1.3 Related work

To the best of our knowledge, the work on differentially private approximation algorithms started with private sketches. Mir, Muthukrishnan, Nikolov, and Wright [25] gave pan-private<sup>7</sup> sketches for heavy hitters. An improved sketch has been recently given by Pagh and Thorup [27]. A private version of the deterministic Misra-Gries sketch [26] for heavy hitters has been recently given by Tětek and Lebeda [22]. Heavy hitters were also investigated in the multi-party computation setting [18], in the local differential privacy setting [3], and using cryptographic assumptions [24, 17, 18].

A sketch for fractional frequency moments  $F_p$  for  $0 \leq p \leq 1$  has been given by Wang, Pinelis, and Song [33]. After releasing this paper, Epasto, Mao, Andres, Mirrokni, Vassilvitskii, and Zhong [15] have given an algorithm general value of  $p$  in the continual release setting. A sketch for differentially private quantiles has been given by Alabi, Ben-Eliezer, and Chaturvedi [1]. A technique for stream sanitization has been given by Kaplan and Stemmer [19]; this work resulted in improved differentially private sketches for approximate quantiles. An approach for differentially privately estimating distances in euclidean spaces using private sketches has been given by Stausholm [31]. A general approach to making linear sketches differentially private was given by Zhao, Qiao, Redberg, Agrawal, Abbadi, and Wang [35].

A recent line of work has shown that many sketches already provide privacy *by themselves* or with only small modifications, without adding any noise. Blocki, Blum, Datta, and Sheffet [6] have shown that the Johnson-Lindenstrauss transform by itself ensures differential privacy. Smith, Song, and Guha Thakurta [30] have shown that this is also the case for the Flajolet-Martin sketch for counting distinct elements and a similar result is known for the LogLog algorithm [9, 11]. This was recently generalized [10] to show that this is not only the case for the two above-mentioned sketches, but in fact for a large class of sketches for counting distinct elements.

As far as sublinear-time algorithms are concerned, Sivasubramaniam, Li, and He [29] have shown a differentially private algorithm that returns a  $2 + \rho$  approximation of the number of edges in a graph in time  $\tilde{O}_{\rho, \epsilon}(\sqrt{n})$ . This has been later improved by Blocki, Grigorescu, and Mukherjee [5] to  $1 + \rho$  approximation in the same complexity. In that paper, the authors also give differentially private sublinear algorithms for approximate maximum matching and vertex cover. A sublinear time algorithm for estimating the median was recently discovered [8].

<sup>7</sup> An algorithm on an input stream is said to be pan-private if releasing the internal state of the algorithm at any point in the computation is differentially private. It is a strictly stronger notation than differential privacy of the output.

## Concurrent work

The problem of differentially private randomized approximation algorithms has been explored independently of this work by Blocki, Grigorescu, Mukherjee, and Zhou [7]. The techniques used in [7] differ significantly from those used in this paper. Specifically, in [7], the authors set the failure probability of the algorithm to be  $\leq \delta$  (for example by probability amplification), thus limiting the global sensitivity of the algorithm up to an event of probability  $\leq \delta$ . This allows them to rely on the standard result for getting differential privacy based on global sensitivity<sup>8</sup>. Specifically, if the algorithm has complexity  $T(n, \rho)$  and one uses probability amplification, then the approach of [7] gives an  $(\epsilon, \delta)$ -differential privacy in complexity  $O(T(n, \epsilon\rho) \log \delta^{-1})$ . They then show that one can post-process the output of the algorithm to achieve pure differential privacy.

In this paper, instead of relying just on global sensitivity, we instead prove privacy from first principles. At the cost of assuming that the error is sufficiently concentrated, we show that the probability amplification step is not needed, allowing us to get  $\epsilon$ -differential privacy in the better complexity of  $T(n, \epsilon\rho)$ . Our approach allows us to give more efficient algorithms than Blocki, Grigorescu, Mukherjee, and Zhou [7] for several problems: estimating the average degree of a graph, estimating the size of a maximum matching, and estimating the number of connected components.

## 2 Preliminaries

### 2.1 Differential privacy

Throughout the paper, we assume that we have a symmetric “neighbor” relation  $\sim$  on the set of all possible datasets. Intuitively speaking, in the case when we have a database of users, this should correspond to two datasets being the same except for the data of one user whose privacy we are trying to protect.

► **Definition 1** ([12]). *A randomized algorithm  $M$  with range  $S$  is  $\epsilon$ -differentially private if for any measurable  $T \subseteq S$ , it holds for any  $x \sim x'$  for a symmetric “neighbor” relation  $\sim$ , that*

$$e^{-\epsilon} \leq \frac{\mathbb{P}[M(x) \in T]}{\mathbb{P}[M(x') \in T]} \leq e^{\epsilon}$$

This definition is commonly relaxed to a notion called approximate differential privacy, with the above notion then being called pure privacy. In this paper, we will focus only on pure differential privacy.

If the output of  $M$  is a continuous random variable, then it is sufficient to prove that for any  $y$  and  $x \sim x'$  it holds  $e^{-\epsilon} \leq f_{M(x)}(y)/f_{M(x')}(y) \leq e^{\epsilon}$ , where  $f_X$  for  $X$  being a continuous random variable is the probability density function of  $X$ .

The global sensitivity [13] of a function  $g$  is defined as

$$\sup_{x \sim x'} |g(x) - g(x')|.$$

The authors have shown that if  $g$  has global sensitivity  $\Delta$ , then adding  $\text{Laplace}(\Delta/\epsilon)$  provides  $\epsilon$ -differential privacy.

<sup>8</sup> They also consider functions with low smoothed sensitivity instead of just low global sensitivity; we do not consider that in this paper.

In the context of graph problems, one often speaks of a mechanism being edge-differentially private, or node-differentially private. These terms refer to the relation  $\sim$  that is used. In the case of node-differential privacy, we have  $G \sim G'$  iff one can get  $G$  from  $G'$  by deleting one vertex and the incident edges, or the other way around. In the case of edge-differential privacy, we have  $G \sim G'$  iff one can get  $G$  from  $G'$  by deleting one edge, or the other way around.

## 2.2 Probability theory

If  $D$  is a distribution, we use  $D^{\otimes k}$  for  $k$  being a natural number, to denote the  $k$ -fold product distribution of  $D$ . For a random variable  $Z$ , we denote by  $F_Z$  its cumulative distribution function. We denote by  $F_Z^{-1}(p) = \inf\{x \in \mathbb{R} : F_Z(x) \geq p\}$  its generalized inverse. It holds that  $F_Z^{-1}(u)$  has the same distribution as  $Z$  for  $u \sim \text{Unif}(0, 1)$  [23]. We will need the following claim.

► **Fact 2** ([16]). *Let  $X, Y$  be independent random variables in  $\mathbb{R}^k$ , and assume  $Y$  has a probability density function (pdf)  $f_Y(z)$ . Then the pdf of  $X + Y$  is  $f_{X+Y}(z) = \mathbb{E}[f_Y(z - X)]$ .*

### Concentration of measure

The notions of *subexponential* random variables and *subexponential diameter*  $\sigma_{se}[X]$  are central to concentration of measure. There are several different definitions for  $\sigma_{se}[X]$ , that differ by constant factors. See for example [32, Chapter 2] for an exposition of the various definitions. In this paper, one of the definitions is especially suitable for the way we use it in our proofs, and that is the definition that we use.

► **Definition 3.** *Let  $X$  be a real random variable. We define the subexponential diameter of  $X$ , denoted by  $\sigma_{se}[X]$  as the smallest values for which for any  $t > 0$  holds*

$$\mathbb{P}[|X| \geq t] \leq 2 \exp(-t/\sigma_{se}[X])$$

A random variable  $X$  is said to be subexponential if  $\sigma_{se}[X] < \infty$ .

It holds that  $\sigma_{se}[cX] = c\sigma_{se}[X]$ . It also holds that

► **Fact 4.** *For  $X$  being a random variable and  $c \geq 0$ , it holds that  $\sigma_{se}[X+c] \leq \sigma_{se}[X] + c/\log 2$ .*

**Proof.** We can bound  $\mathbb{P}[X + c \geq t] \leq \min(1, 2e^{-(t-c)/\sigma_{se}[X]}) \leq \min(1, 2e^{-t/(\sigma_{se}[X]+c/\log 2)})$ , thus implying the claim. ◀

Finally, the following is a standard claim, but we need the constant factor, which is specific to the definition of  $\sigma_{se}$  that we are using. We thus give a proof in the full version of this paper, based on the standard proof of triangle inequality for Orlicz norms.

► **Lemma 5.** *Let us have real random variables  $X_1, \dots, X_k$ . It holds*

$$\sigma_{se}\left[\sum_{i=1}^k X_i\right] \leq 3 \sum_{i=1}^k \sigma_{se}[X_i]$$

### 3 Algorithms with subexponential error

In this section, we show how algorithms, whose error has a small subexponential diameter, can be made differentially private. We start by proving a technical lemma. We will later bound the ratios between probability densities of our mechanism's answers by the exponential expectations that we now bound and, finally, we will use that to prove privacy in Theorem 9, which is the main theorem of this section.

Note that while the second of the two inequalities is standard, the first one is not. Our proof does not follow the strategy of the standard proof of the second inequality, which is based on a Taylor expansion and does not seem to straightforwardly apply to the first inequality.

► **Lemma 6.** *Suppose  $X$  is a random variable with subexponential diameter  $\Delta \leq 1/2$ . It holds  $\mathbb{E}[e^{-|X|}] \geq \frac{2^{-\Delta}}{1+\Delta} \geq e^{-(1+\log 2)\Delta}$  and  $\mathbb{E}[e^{|X|}] \leq \frac{2^\Delta}{1-\Delta} \leq e^{3\log(2)\Delta}$ .*

**Proof.** Since  $X$  is subexponential with diameter  $\Delta$ , it holds that  $\mathbb{P}[|X| \geq z] \leq 2e^{-z/\Delta}$ . Therefore,  $F_{|X|}^{-1}(u) \leq -\Delta \log(\frac{1-u}{2})$ . We use the fact that for  $u \sim \text{Unif}(0, 1)$ , the random variable  $F_{|X|}^{-1}(u)$  has the same distribution as  $|X|$ . We can now bound

$$\begin{aligned}
 \mathbb{E}[e^{-|X|}] &= E_u[e^{-F_{|X|}^{-1}(u)}] \\
 &\geq E_u[e^{\Delta \log(\frac{1-u}{2})}] \\
 &= 2^{-\Delta} E_u[(1-u)^\Delta] \\
 &= 2^{-\Delta} \int_0^1 (1-u)^\Delta du \\
 &= 2^{-\Delta} \left[ -\frac{(1-u)^{\Delta+1}}{\Delta+1} \right]_{u=0}^1 \\
 &= \frac{2^{-\Delta}}{1+\Delta} \geq e^{-(1+\log 2)\Delta}
 \end{aligned} \tag{1}$$

where the last inequality holds because we can equivalently write  $2^{-\Delta}/(\Delta+1) \geq (2e)^{-\Delta}$ , which simplifies to  $e^\Delta \geq 1+\Delta$ , which is a standard inequality. Similarly, we can bound  $\mathbb{E}[e^{|X|}]$  as follows.

$$\begin{aligned}
 \mathbb{E}[e^{|X|}] &= E_u[e^{F_{|X|}^{-1}(u)}] \\
 &\leq E_u[e^{-\Delta \log(\frac{1-u}{2})}] \\
 &= \frac{2^\Delta}{1-\Delta} \leq e^{3\log(2)\Delta}
 \end{aligned}$$

where the second equality is by substituting  $\Delta$  with  $-\Delta$  in (1) (since as we have shown, (1) is equal to  $2^{-\Delta}/(1+\Delta)$ ). We have here used that  $\Delta < 1$  (otherwise the final expression may not be defined). The last inequality can be shown as follows: we take the ratio of the two sides resulting in  $h(\Delta) = 4^\Delta(1-\Delta)$  and we show  $h(\Delta) \geq 1$  for  $0 \leq \Delta \leq 1/2$ . The function  $h$  is concave (the second derivative is  $-2 \cdot 2^{2\Delta} \log 4 + 2^{2\Delta} \log^2 4$ , which can be easily seen to be negative), meaning that it is sufficient to check that the inequality holds at the endpoints of the interval  $[0, 1/2]$ : that  $h(0) \geq 1$  and  $h(1/2) \geq 1$ . One can easily check this holds. ◀

We are now ready to prove a lemma about the ratio of the density function of a Laplace and of Laplace shifted by a subexponential random variable. We will then use this to prove differential privacy. Note that the random variables in the lemma do not have to be independent.

► **Lemma 7.** *Let  $X_1, \dots, X_k$  be random variables with subexponential diameter at most  $\Delta$  and let  $X = (X_1, \dots, X_k)$ . Let  $Y \sim \text{Lap}^{\otimes k}(k\Delta/\epsilon)$  for  $\epsilon \leq 1/6$ . Consider  $y \in \mathbb{R}^k$ . It holds  $e^{-3(1+\log 2)\epsilon} \leq f_{X+Y}(y)/f_Y(y) \leq e^{9\log(2)\epsilon}$ . Moreover, if  $k = 1$  and  $\epsilon \leq 1/2$ , it holds  $e^{-(1+\log 2)\epsilon} \leq f_{X+Y}(y)/f_Y(x) \leq e^{3\log(2)\epsilon}$ .*

**Proof.** By Fact 2, we have

$$f_{X+Y}(y) = \mathbb{E}[f_Y(y - X)] = \left(\frac{\epsilon}{2k\Delta}\right)^k \mathbb{E}[\exp(-\frac{\epsilon}{k\Delta}\|X - y\|_1)]$$

For the sake of brevity, we let  $\gamma = \left(\frac{\epsilon}{2k\Delta}\right)^k$ . We may bound  $\|X - y\|_1 \leq \|X\|_1 + \|y\|_1$  and  $\|X - y\|_1 \geq \|y\|_1 - \|X\|_1$ . This allows us to bound

$$\begin{aligned} f_{X+Y}(y) &= \gamma E \left[ \exp\left(\frac{-\epsilon\|X - y\|_1}{k\Delta}\right) \right] \\ &\geq \gamma E \left[ \exp\left(\frac{-\epsilon(\|X\|_1 + \|y\|_1)}{k\Delta}\right) \right] \\ &= \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) E \left[ \exp\left(\frac{-\epsilon\|X\|_1}{k\Delta}\right) \right] \\ &\geq \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) \exp(-3(1 + \log 2)\epsilon) \end{aligned} \tag{2}$$

where the last inequality is by Lemma 6; we used that  $\frac{\epsilon\|X\|_1}{k\Delta}$  has subexponential diameter  $\leq 3\epsilon \leq 1/2$ , since we have by Lemma 5 that  $\sigma_{se}[\|X\|] \leq 3k\Delta$ . In the case  $k = 1$ , we simply have that  $\sigma_{se}[\frac{\epsilon\|X\|_1}{k\Delta}] \leq \epsilon$ , in which case the final bound on (2) is

$$\geq \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) \exp(-(1 + \log 2)\epsilon)$$

At the same time,  $f_Y(y) = \gamma \exp(-\frac{\epsilon\|y\|_1}{k\Delta})$  and thus

$$\begin{aligned} \frac{f_{X+Y}(y)}{f_Y(y)} &\geq e^{-3(1+\log 2)\epsilon} && \text{if } k > 1 \\ \frac{f_{X+Y}(y)}{f_Y(y)} &\geq e^{-(1+\log 2)\epsilon} && \text{if } k = 1 \end{aligned}$$

Similarly, we can bound

$$\begin{aligned} f_{X+Y}(y) &= \gamma E \left[ \exp\left(\frac{-\epsilon\|X - y\|_1}{k\Delta}\right) \right] \\ &\leq \gamma E \left[ \exp\left(\frac{-\epsilon(\|y\|_1 - \|X\|_1)}{k\Delta}\right) \right] \\ &= \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) E \left[ \exp\left(\frac{\epsilon\|X\|_1}{k\Delta}\right) \right] \\ &\leq \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) \exp(9\log(2)\epsilon) \end{aligned}$$

in the case  $k > 1$  and

$$f_{X+Y}(y) \leq \gamma \exp\left(\frac{-\epsilon\|y\|_1}{k\Delta}\right) \exp(3\log(2)\epsilon)$$

in the case  $k = 1$ . Thus, we have

$$\begin{aligned} \frac{f_{X+Y}(y)}{f_Y(y)} &\leq e^{9 \log(2)\epsilon} && \text{if } k > 1 \\ \frac{f_{X+Y}(y)}{f_Y(y)} &\leq e^{3 \log(2)\epsilon} && \text{if } k = 1 \end{aligned} \quad \blacktriangleleft$$

We now state a useful technical lemma; the proof appears in the full version. This lemma states that in general, if we have a countable number of mechanisms and releasing any fixed  $k$  of them is differentially private, then picking the mechanisms adaptively will not violate differential privacy. The proof roughly follows the outline of the proof of adaptive composition [28]. In what follows, we again use the subscript  $\cdot_r$  to denote a random bitstring used as the source of randomness of the mechanisms.

► **Lemma 8.** *Let us have a countable number of mechanisms  $M_{1,r}, \dots$ , such that releasing the value  $(M_{i_1,r}(D), \dots, M_{i_k,r}(D))$  is  $\epsilon$ -differentially private for any fixed  $i_1, \dots, i_k \in [n]$ . Then the mechanism  $(M_{j_1,r}(D), \dots, M_{j_k,r}(D))$  is  $\epsilon$ -differentially private for  $j_1, \dots, j_k \in [n]$  such that  $j_\ell$  for  $\ell \in [k]$  is drawn from a distribution which is a function of  $M_{j_1,r}(D), \dots, M_{j_{\ell-1},r}(D)$ .*

We are now ready to prove the main theorem of this section. In what follows, we denote by  $A_r(\cdot)$  the algorithm  $A$  executed with randomness  $r$ . We formalize the multiple-query setting as having one algorithm which takes as part of its input a query. This differs somewhat from the presentation in the introduction which assumed we have a sequence of algorithms, which we chose there as it required less notation. Note also that while we are not assuming that the algorithm does not know which phase it is (the value of  $i$ ), we may without loss of generality assume this is passed as part of the query. Note that the condition on the queries  $x_1, \dots, x_k$  below simply states that the queries can be chosen adaptively based on the released values, and that they do not have to be deterministic. Note also that the randomness  $r$  must not be released, as the privacy also relies on that randomness.

► **Theorem 9.** *Let us have an algorithm  $A(D, x)$  for a database  $D$  and a query  $x \in U$ , where  $U$  is a countable universe. Assume there exists a function  $g(D, x)$  with its global sensitivity w.r.t.  $D$  being  $\leq \Delta_1$  for any  $x$ , and such that  $\sigma_{se}[A(D, x) - g(D, x)] \leq \Delta_2$  for any  $D, x$ .*

*Pick at random  $Y_i \sim \text{Laplace}(c(\Delta_1 + \Delta_2)k/\epsilon)$  for  $c = 3 + 12 \log 2$  and for  $\epsilon \leq c/6$  and pick  $r$  independently uniformly on  $\{0, 1\}^\infty$ . Then for queries  $x_1, \dots, x_k \in U$  where the query  $x_i$  is drawn from a distribution that is a function of  $A_r(D, x_1) + Y_1, \dots, A_r(D, x_{i-1}) + Y_{i-1}$ , releasing  $(A_r(D, x_1) + Y_1, \dots, A_r(D, x_k) + Y_k)$  is  $\epsilon$ -differentially private, with the privacy also being over the randomness of  $r$ .*

*If  $k = 1$ , then  $c = 1 + 4 \log 2$  and  $\epsilon \leq c/2$  is sufficient.*

**Proof.** Let us have two neighboring databases  $D_1, D_2$ . Let  $Y = (Y_1, \dots, Y_k)$ , and for  $x = (x_1, \dots, x_k)$ , let  $A'(D, x) = (A_r(D, x_1) + Y_1, \dots, A_r(D, x_k) + Y_k)$  for  $r$  uniform on  $\{0, 1\}^\infty$ . Similarly, let  $g(D, x) = (g(D, x_1), \dots, g(D, x_k))$ . We prove that for any fixed (non-adaptive) queries  $x = (x_1, \dots, x_k)$  it holds  $f_{A'(D_1, x)}(y)/f_{A'(D_2, x)}(y) \leq e^\epsilon$ . If we prove this, the theorem follows: the inequality  $f_{A'(D_1, x)}(y)/f_{A'(D_2, x)}(y) \geq e^{-\epsilon}$  holds by symmetry and these bounds together imply  $\epsilon$ -differential privacy. Lemma 8 then imply that  $A'$  is differentially private even for adaptive queries.

Let  $R_1 = A'(D_1, x) - g(D_1, x)$  and  $R_2 = A'(D_2, x) - g(D_1, x)$  (note the asymmetry in the definitions). We are assuming  $R_1$  has subexponential diameter  $\leq \Delta_2$ . By Fact 4, it holds that  $R_2$  has subexponential diameter  $\leq \Delta_1/\log(2) + \Delta_2$ . We now have from Lemma 7 the following bounds

$$\begin{aligned}\frac{f_{g(D_1,x)+R_1+Y}(y)}{f_{g(D_1,x)+Y}(y)} &= \frac{f_{R_1+Y}(y-g(D_1,x))}{f_Y(y-g(D_1,x))} \leq e^{9\log(2)\epsilon/c} \\ \frac{f_{g(D_2,x)+R_2+Y}(y)}{f_{g(D_1,x)+Y}(y)} &= \frac{f_{R_2+Y}(y-g(D_2,x))}{f_Y(y-g(D_1,x))} \geq e^{-3(1+\log 2)\epsilon/c}\end{aligned}$$

which in turn allows us to bound

$$f_{A'(D_1,x)}(y)/f_{A'(D_2,x)}(y) = \frac{f_{g(D_1,x)+R_1+Y}(x)}{f_{g(D_1,x)+Y}(x)} \cdot \frac{f_{g(D_1)+Y}}{f_{g(D_2)+R_2+Y}(x)} \leq e^{(3+12\log 2)\epsilon/c} = e^\epsilon$$

If  $k = 1$ , the same computation gives the desired bound for  $c = 1 + 4\log 2$ , since Lemma 7 in that case gives

$$\begin{aligned}\frac{f_{g(D_1,x)+R_1+Y}(y)}{f_{g(D_1,x)+Y}(y)} &\leq e^{3\log(2)\epsilon/c} \\ \frac{f_{g(D_2,x)+R_2+Y}(y)}{f_{g(D_1,x)+Y}(y)} &\geq e^{-(1+\log 2)\epsilon/c}\end{aligned}$$

which again results in the bound  $f_{A'(D_1,x)}(y)/f_{A'(D_2,x)}(y) \leq e^\epsilon$ . ◀

#### 4 Algorithms with bounded mean error

In this section, we show a weaker version of Theorem 9 that only requires that the error has some number of bounded moments, instead of requiring that it is subexponential. We start by defining the distribution that we will use in our additive noise mechanism.

► **Definition 10.** *Zero-symmetric Pareto distribution with shape parameter  $\alpha > 1$  and scale parameter  $s > 0$ , denoted  $ZSPareto_\alpha(s)$ , is defined by the PDF*

$$\frac{1}{2s}(\alpha - 1)(|x|/s + 1)^{-\alpha}$$

Before we can prove the main theorem of this section, we need the following technical lemma. The proof is rather technical and it appears in the full version of this paper.

► **Lemma 11.** *Let us have any  $0 \leq \epsilon \leq 1$ ,  $\alpha > 1$ , and  $x \geq 0$ . It holds*

$$\int_0^1 \min\left((1+x)^\alpha, \left|1 - \frac{(1-2^{-1/\alpha})\epsilon}{(1+x)(1-u)^{1/\alpha}}\right|^{-\alpha}\right) du \leq 1 + \frac{2\alpha-1}{\alpha-1}\epsilon \quad (3)$$

We are now ready to prove a lemma which bound the privacy loss of our mechanism. In what follows, we use the notation  $\|X\|_p$  for a random variable  $X$  to denote the  $L_p$  norm  $\sqrt[p]{\mathbb{E}[|X|^p]}$ .

► **Lemma 12.** *Let  $X$  be a real random variable such that  $\|X\|_\alpha \leq \Delta$  and let  $Y \sim ZSPareto_\alpha(s)$  for  $s = (1 - 2^{-1/\alpha})^{-1}\Delta/\epsilon$  for  $0 \leq \epsilon \leq 1$  and  $\alpha > 1$ . Consider  $y \in \mathbb{R}$ . It holds  $e^{-(1-2^{-1/\alpha})\alpha\epsilon} \leq f_{X+Y}(y)/f_Y(y) \leq e^{\frac{2\alpha-1}{\alpha-1}\epsilon}$ .*

**Proof.** Since  $\|X\|_\alpha \leq \Delta$ , it holds by the higher-order Chebyshev inequality<sup>9</sup> that  $\mathbb{P}[|X| \geq z] \leq \Delta^\alpha/z^\alpha$ . Therefore,  $F_{|X|}^{-1}(u) \leq \Delta/\sqrt[\alpha]{1-u}$ . We will use that  $|y - X| \leq |y| + |X|$  and later below also that  $|y - X| \geq \max(0, |y| - |X|)$ . We start with the simpler case of proving a lower bound.

$$\begin{aligned}
\frac{f_{X+Y}(y)}{f_Y(y)} &= \frac{\mathbb{E}[f_Y(y - X)]}{(|y|/s + 1)^{-\alpha}} & (4) \\
&= \frac{\mathbb{E}[ (|y - X|/s + 1)^{-\alpha} ]}{(|y|/s + 1)^{-\alpha}} \\
&\geq E \left[ \left( \frac{|y|/s + |X|/s + 1}{|y|/s + 1} \right)^{-\alpha} \right] \\
&= E \left[ \left( 1 + \frac{|X|/s}{|y|/s + 1} \right)^{-\alpha} \right] \\
&\geq \mathbb{E}[(1 + |X|/s)^{-\alpha}] \\
&\geq \mathbb{E}[1 - \alpha|X|/s] \\
&= 1 - \alpha\mathbb{E}[|X|]/s \\
&\geq 1 - \frac{\alpha(1 - 2^{-1/\alpha})\Delta}{\Delta} \epsilon \geq e^{-(1-2^{-1/\alpha})\alpha\epsilon} & (5)
\end{aligned}$$

where (4) holds by Fact 2, and (5) uses that  $\mathbb{E}[|X|] = \|X\|_1 \leq \|X\|_\alpha \leq \Delta$ . We now show the upper bound part. We prove an upper bound in terms of the integral which we have bounded in Lemma 11.

$$\begin{aligned}
\frac{f_{X+Y}(y)}{f_Y(y)} &= \frac{\mathbb{E}[ (|y - X|/s + 1)^{-\alpha} ]}{(|y|/s + 1)^{-\alpha}} \\
&\leq E \left[ \min \left( (1 + |y|/s)^\alpha, \left| \frac{|y|/s - |X|/s + 1}{|y|/s + 1} \right|^{-\alpha} \right) \right] \\
&= E \left[ \min \left( (1 + |y|/s)^\alpha, \left| 1 - \frac{|X|/s}{|y|/s + 1} \right|^{-\alpha} \right) \right] \\
&= E_u \left[ \min \left( (1 + |y|/s)^\alpha, \left| 1 - \frac{F_{|X|}^{-1}(u)/s}{|y|/s + 1} \right|^{-\alpha} \right) \right] \\
&\leq E_u \left[ \min \left( (1 + |y|/s)^\alpha, \left| 1 - \frac{(1 - 2^{-1/\alpha})^{-1}\epsilon}{(|y|/s + 1)(1 - u)^{1/\alpha}} \right|^{-\alpha} \right) \right] \\
&= \int_0^1 \min \left( (1 + |y|/s)^\alpha, \left| 1 - \frac{(1 - 2^{-1/\alpha})^{-1}\epsilon}{(|y|/s + 1)(1 - u)^{1/\alpha}} \right|^{-\alpha} \right) du \\
&\leq 1 + \frac{2\alpha - 1}{\alpha - 1} \epsilon \leq e^{\frac{2\alpha - 1}{\alpha - 1} \epsilon}
\end{aligned}$$

where we have proven the inequality second to last in Lemma 11. ◀

We are now ready to prove the main theorem of this section.

<sup>9</sup> The higher-order Chebyshev inequality states that for  $X$  being a real random variable, it holds  $\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq t^\alpha/\mathbb{E}[|X - \mathbb{E}[X]|^\alpha]$  for any  $\alpha \geq 0$ .



► **Theorem 13.** *Let us have an algorithm  $A(D)$  such that there exists a function  $g(D)$  with global sensitivity  $\leq \Delta_1$  w.r.t.  $D$  for which for any input  $D$ , it holds  $\mathbb{E}[|A(D) - g(D)|^\alpha] \leq \Delta_2^\alpha$  for some  $\alpha > 1$ . Let  $Y \sim ZSPareto_\alpha(c(\Delta_1 + \Delta_2)/\epsilon)$  for  $c = \alpha + 2 + 1/(\alpha - 1)$  and  $\epsilon \leq c$ , independent of the randomness of  $A$ ; then  $A(D) + Y$  is  $\epsilon$ -differentially private with respect to  $D$ .*

**Proof.** This proof follows the strategy of the proof of Theorem 9. Let us have two neighboring databases  $D_1, D_2$ . We again prove that for any  $y$ , it holds  $f_{A(D_1)}(y)/f_{A(D_2)}(y) \leq e^\epsilon$ ; this implies the theorem. We also again set  $R_1 = A(D_1) - g(D_1)$  and  $R_2 = A(D_2) - g(D_1)$ . We are assuming  $\|R_1\|_\alpha \leq \Delta_2$  and by the triangle inequality, we have that  $\|R_2\|_\alpha \leq \Delta_1 + \Delta_2$ . Therefore, we have

$$\begin{aligned} \frac{f_{g(D_1)+R_1+Y}(y)}{f_{g(D_1)+Y}(y)} &= \frac{f_{R_1+Y}(y - g(D_1))}{f_Y(y - g(D_1))} \leq \exp\left(\frac{2\alpha - 1}{(\alpha - 1)c}\epsilon\right) \\ \frac{f_{g(D_2)+R_1+Y}(y)}{f_{g(D_1)+Y}(y)} &= \frac{f_{R_1+Y}(y - g(D_2))}{f_Y(y - g(D_1))} \geq \exp\left(-(1 - 2^{-1/\alpha})\alpha\epsilon/c\right) \end{aligned}$$

which in turn allows us to bound

$$\begin{aligned} f_{A(D_1)+Y}(y)/f_{A(D_2)+Y}(y) &= \frac{f_{g(D_1)+R_1+Y}(y)}{f_{g(D_1)+Y}(y)} \cdot \frac{f_{g(D_1)+Y}(y)}{f_{g(D_2)+R_1+Y}(y)} \\ &\leq \exp\left(\left(\frac{2\alpha - 1}{\alpha - 1} + (1 - 2^{-1/\alpha})\alpha\right)\epsilon/c\right) \leq e^\epsilon \end{aligned}$$

where we now argue the last inequality; that will conclude the proof. If we set  $c = 2 + 1/(\alpha - 1) + \alpha - 2^{-1/\alpha}\alpha$ , the last inequality would be an equality. By monotonicity, it is thus sufficient to prove that  $2 + 1/(\alpha - 1) + \alpha - 2^{-1/\alpha}\alpha \leq 2 + \log 2 + 1/(\alpha - 1)$ . This is equivalent to  $2^{-1/\alpha}\alpha \geq \alpha - \log 2$ , which in turn can be re-written as  $2^{-1/\alpha} \geq 1 - \log(2)/\alpha$ . This follows from the inequality  $e^x \geq 1 + x$  for  $x = -\log(2)/\alpha$ . ◀

## 5 Implications of our results

In this section, we give several implications of Theorem 9 and Theorem 13. This list is by no means meant to be exhaustive. We start with the more straightforward applications and focus on the more involved ones later, with one part being deferred to the full version of this paper.

Recall that, as we discussed, the goal in the sublinear setting is not to simply add small amount of noise, but rather to achieve a given level of error as efficiently as possible while guaranteeing differential privacy. This is so because in this setting, the amount of error coming from the algorithm not being exact tends to be much greater than the amount of noise needed to achieve privacy when not subject to having limited resources.

### 5.1 The general approach

In all applications, we take a known algorithm for a given problem, and use either Theorem 9 or Theorem 13 to argue that adding noise to the algorithm's answer ensures privacy.

Assume the original algorithm had complexity  $T(n, \rho)$  and assume for example that the error is of magnitude  $\rho n$ , namely that for error  $R$ , it holds  $E[R^2]^{1/2} \leq O(\rho n)$  (this can be generalized to  $\leq O(\rho f(x))$  for  $x$  being the input and  $f$  being any function). We run the algorithm with parameter  $\rho' = \epsilon\rho$  and add noise of magnitude  $O(\rho n)$  (more generally

$O(\rho f(x))$ ). By Theorem 13 with  $\alpha = 2$ , as long as the approximated function's sensitivity is  $\Delta \leq \epsilon \rho n$ , this is  $\epsilon$ -differentially private<sup>10</sup>. At the same time, the error is  $\leq O(\rho n)$  with arbitrarily high constant probability. The time complexity is  $T(n, \epsilon \rho)$ .

If we want to achieve a failure probability of  $\beta$ , we run this algorithm  $\Theta(\log \beta^{-1})$  times and take the median. By a standard probability amplification argument, the success probability will be as desired. To achieve  $\epsilon$ -differential privacy by composition, we have to divide the privacy budget between the runs, resulting in complexity  $O(T(n, \epsilon \rho / \log \beta^{-1}) \log \beta^{-1})$ . This can be summarized (and generalized with the general function  $f(x)$ ) as follows:

► **Lemma 14.** *Suppose there is an algorithm approximating a function  $g$  with global sensitivity  $\Delta$  such that  $E[(A(x) - g(x))^2]^{1/2} \leq \rho f(x)$  for some function  $f$  with time/space/query complexity  $T(n, \rho)$ . Then for  $\epsilon \leq O(1)$  there exists an  $\epsilon$ -differentially private algorithm  $A'$  such that when  $\epsilon \rho \geq \Omega(\Delta/f(x))$ , it holds  $P[|A'(x) - g(x)| > \rho f(x)] \leq \beta$  and that has complexity  $O(T(n, \epsilon \rho / \log \beta^{-1}) \log \beta^{-1})$ .*

A more efficient approach for decreasing failure probability exists in the case of subexponential error. Assume the same setting as above, except that the error's subexponential diameter is  $\rho n$  instead having only moment bounds (like above, we can generalize to  $\rho f(x)$  instead of  $\rho n$ ). We run the algorithm with parameter  $\rho' = \epsilon \rho / \log \beta^{-1}$  and add noise of magnitude  $\Theta(\rho n / \log \beta^{-1})$ . This algorithm is  $\epsilon$ -differentially private by Theorem 9, as long as  $\Delta \leq \epsilon \rho n$ . The noise has subexponential diameter  $O(\rho' n)$  and by Lemma 5, the total error will up to a constant have the same subexponential diameter. By the definition of subexponential diameter, the probability that the error is  $\geq \Theta(\rho n)$  is  $\leq \beta$  as desired. This results in complexity  $O(T(n, \epsilon \rho / \log \beta^{-1}))$  saving us one  $\log \beta^{-1}$  factor. We can summarize this as

► **Lemma 15.** *Suppose there is an algorithm approximating a function  $g$  with global sensitivity  $\Delta$  such that  $\sigma_{se}[A(x) - g(x)] \leq \rho f(x)$  for some function  $f$  with time/space/query complexity  $T(n, \rho)$ . Then for  $\epsilon \leq O(1)$ , there exists an  $\epsilon$ -differentially private algorithm  $A'$  such that when  $\epsilon \rho \geq \Omega(\Delta/f(x))$ , it holds  $P[|A'(x) - g(x)| > \rho f(x)] \leq \beta$  and that has complexity  $O(T(n, \epsilon \rho / \log \beta^{-1}))$ .*

We are now ready to give private algorithms for specific problems.

## 5.2 Frequency moment $F_2$

In their seminal paper, Alon, Matias, and Szegedy [2] show a sketch that allows one to estimate the  $F_2$  frequency moment, defined as  $F_2(x_1, \dots, x_n) = \sum_{i=1}^n x_i^2$ . In the streaming setting, the vector  $x_1, \dots, x_n$  is given through a stream of updates  $y_1, \dots, y_k$  of the form  $y_j = (\ell_j, D_j)$  where  $D$  can be negative and we define  $x_i = \sum_{j=1}^k I[\ell_j = i] D_j$ . Two inputs are then adjacent if they differ in one value  $y_j$  for some  $j$ . The algorithm from [2] uses space  $O(\frac{1}{\rho^2})$  and has mean squared error of  $\leq \rho^2 F_2^2 \leq \rho^2 n^4$ . The sensitivity of the  $F_2$  moment is  $n$ . This implies the following

► **Corollary 16.** *For  $\epsilon \leq O(1)$  and  $\rho \leq 1/(\epsilon n)$ , there is an  $\epsilon$ -differentially private algorithm that returns an additive  $\pm \rho n^2$  approximation of the frequency moment  $F_2$  with probability  $1 - \beta$ , and has space complexity  $O(\frac{\log^3 \beta^{-1}}{\rho^2 \epsilon^2})$ .*

<sup>10</sup>This upper bound on the sensitivity ensures that  $\Delta_2$  in Theorem 13 dominates.

This improves upon the concurrent work [7] which gives an algorithm with complexity  $O(\frac{\log^4 n}{\rho^2 \epsilon^2})$ . Shortly after releasing this paper, an approach also appeared with the incomparable complexity of  $O(\log(n) \log^3 \beta^{-1} / \rho^2)$  was shown (for  $\epsilon$  being not too small) that also has multiplicative approximation guarantees and works (with some loss in the complexity) in the continual release setting [15]. The setting of  $F_p$  for  $p \in [0, 1]$  has been considered in [33].

### 5.3 Connected components

An algorithm is known [4] that returns an estimate  $\hat{c}$  of the number of connected components  $c$  of a simple graph in time  $O(\frac{1}{\rho^2} \log \frac{1}{\rho})$  and has mean squared error  $\mathbb{E}[(\hat{c} - c)^2] \leq \rho^2 n^2$ . At the same time, the number of connected components has global sensitivity 1 with respect to edge additions/deletions. This gives us the following

► **Corollary 17.** *For  $\epsilon \leq O(1)$  and  $\rho \leq 1/(en)$ , there is an  $\epsilon$ -edge-differentially private algorithm that returns an additive  $\pm \rho n$  approximation of the number connected components with probability  $1 - \beta$ , and has complexity  $O(\frac{\log^3 \beta^{-1}}{\rho^2 \epsilon^2} \log \frac{\log \beta^{-1}}{\rho \epsilon})$ .*

No private sublinear-complexity algorithm for estimating the number of connected components was previously known.

### 5.4 Maximum matching

Yoshida, Yamamoto, and Ito [34] show an algorithm that can approximate the size of the maximum matching to within multiplicative  $1 + \rho$  in time  $d^{O(1/\rho^2)} (1/\rho)^{O(1/\rho)}$  for  $d$  being the maximum degree of the input graph. It works by implementing an oracle for a matching of size within factor  $1 + \rho/2$  of the maximum matching; for a specified vertex, this oracle answers whether the vertex is matched in the oracle's matching. The algorithm then samples  $\Theta(1/\rho^2)$  vertices and checks the fraction that is matched in the oracle's matching. The error coming from the oracle is  $\leq \rho n/2$  in the worst case and thus has subexponential diameter  $O(\rho n)$ . The error coming from the sampling has subexponential diameter  $O(\rho n)$  by the Hoeffding inequality. By Lemma 5, the subexponential diameter of the error is thus  $O(\rho n)$ . At the same time, the global sensitivity of the maximum matching size is  $\leq 1$  with respect to the removal of one vertex. This gives us the following

► **Corollary 18.** *For  $\epsilon \leq O(1)$  and  $\rho \leq 1/(en)$ , there is an  $\epsilon$ -node-differentially-private algorithm that returns an additive  $\pm \rho n$  approximation of the maximum matching size with probability  $1 - \beta$  in time  $d^{O(\log^2(\beta^{-1})/(\rho^2 \epsilon^2))} / (\rho \epsilon)^{O(\log(\beta^{-1})/(\rho \epsilon))}$ .*

Together with the concurrent [7], this solves the open problem posed in [5] where the authors show a hybrid  $(2, \rho n)$  approximation, while we give a purely additive  $\pm \rho n$  approximation.

### 5.5 Rank queries

Karnin, Lang, and Liberty [20] develop a sketch of size  $O(\frac{1}{\rho})$  that allows one to answer rank queries with error with subexponential diameter  $\rho n$ . We show how to use their sketch to answer range queries over an ordered universe. For small number of queries, this improves upon the work of [19] which has a logarithmic dependency on the universe size. This gives us the following corollary.

► **Corollary 19.** *There is a sketch that allows  $\epsilon$ -differentially private algorithm that returns  $k$  rank queries (potentially adaptive) for  $\epsilon \leq O(1)$  with an additive  $\pm \rho n$  error with probability  $1 - \beta$ , and has complexity  $O(\frac{k \log^2(k/\beta)}{\rho \epsilon})$ .*

**Proof.** We use the KLL sketch with error parameter  $\rho' = \rho\epsilon/(k \log(k/\beta))$ . This means that the error has a subexponential diameter of  $\leq \rho\epsilon n/(k \log(k/\beta))$ . Therefore, by Theorem 9, it holds that using for each query a Laplace mechanism with error magnitude  $\Theta(\rho n/\log(k/\beta))$  will result in  $\epsilon$ -differential privacy.

By Lemma 5, the overall subexponential diameter of the error of each answer is  $O(\rho n/\log(k/\beta))$  and therefore the probability of error  $O(\rho n)$  is  $1 - \beta/k$ . By the union bound, the overall success probability is  $\geq 1 - \beta$ .  $\blacktriangleleft$

This is in comparison to the approach of Kaplan and Stemmer [19] which results in space complexity  $O(\frac{\log |U| \log(k/\beta)}{\rho\epsilon})$ , improving by a factor of  $\log |U|$  for constant  $k, \beta$ , where  $U$  is the universe.

## 5.6 Relative approximation sublinear-time algorithms

In the full version of this paper, we show a general theorem that implies that many sublinear-time algorithms that have relative error guarantees can be made differentially private. Specifically, it applies to many algorithms that rely on a commonly used “advice removal” technique [14]. Among other things, this theorem implies the following algorithm. This improves upon the work of Blocki, Grigorescu, and Mukherjee [5] who give an algorithm with complexity  $\tilde{O}_{\epsilon, \rho}(\sqrt{n})$  under the assumption  $m \geq \Omega(n)$ .

► **Corollary 20.** *For  $\epsilon \leq O(1)$  and  $\rho \leq 1/(\epsilon n)$ , there exists an  $\epsilon$ -edge-differentially private algorithm that returns a  $1 + \rho$ -approximation of the average degree of a graph with probability  $1 - \beta$  and has complexity  $O(\frac{n \log^3 \beta^{-1}}{\epsilon^2 \rho^2 \sqrt{m}})$ .*

## 6 Open problems and conjectures

We are convinced that our results are only the beginning of the story and that there are many interesting related open problems. These include using normal noise with approximate/zero-concentrated differential privacy, generalizing our method to input-dependent error magnitudes, improving the constants, lower bounds for answering multiple queries when we have polynomial tails. For a more detailed discussion, see the full version of this paper.

---

### References

- 1 Daniel Alabi, Omri Ben-Eliezer, and Anamay Chaturvedi. Bounded space differentially private quantiles. *arXiv preprint*, 2022. [arXiv:2201.03380](https://arxiv.org/abs/2201.03380).
- 2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, 1996.
- 3 Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. *Advances in Neural Information Processing Systems*, 30, 2017.
- 4 Petra Berenbrink, Bruce Krayenhoff, and Frederik Mallmann-Trenn. Estimating the number of connected components in sublinear time. *Information Processing Letters*, 114(11):639–642, 2014.
- 5 J Blocki, E Grigorescu, and T Mukherjee. Privately estimating graph parameters in sublinear time. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, 2022.
- 6 Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. The johnson-lindenstrauss transform itself preserves differential privacy. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 410–419. IEEE, 2012.

- 7 Jeremiah Blocki, Elena Grigorescu, Tamalika Mukherjee, and Samson Zhou. How to Make Your Approximation Algorithm Private: A Black-Box Differentially-Private Transformation for Tunable Approximation Algorithms of Functions with Low Sensitivity. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:24, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.59.
- 8 Jonas Boehler and Florian Kerschbaum. Secure sublinear time differentially private median computation, February 1 2022. US Patent 11,238,167.
- 9 Seung Geol Choi, Dana Dachman-Soled, Mukul Kulkarni, and Arkady Yerukhimovich. Differentially-private multi-party sketching for large-scale statistics. *Cryptology ePrint Archive*, 2020.
- 10 Charlie Dickens, Justin Thaler, and Daniel Ting. (nearly) all cardinality estimators are differentially private. *arXiv preprint*, 2022. arXiv:2203.15400.
- 11 Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*, pages 605–617. Springer, 2003.
- 12 Cynthia Dwork. Differential Privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- 13 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
- 14 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 722–734, 2018.
- 15 Alessandro Epasto, Jieming Mao, Andres Munoz Medina, Vahab Mirrokni, Sergei Vassilvitskii, and Peilin Zhong. Differentially private continual releases of streaming frequency moment estimations. *arXiv preprint*, 2023. arXiv:2301.05605.
- 16 geetha290krm (<https://math.stackexchange.com/users/1064504/geetha290krm>). Does  $f_{X+Y}(z) = e[f_Y(z - x)]$  hold? Mathematics Stack Exchange, 2022. URL:<https://math.stackexchange.com/q/4544852> (version: 2022-10-04). arXiv:<https://math.stackexchange.com/q/4544852>.
- 17 Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 463–488. Springer, 2021.
- 18 Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. Frequency estimation under multiparty differential privacy: One-shot and streaming. *arXiv preprint*, 2021. arXiv:2104.01808.
- 19 Haim Kaplan and Uri Stemmer. A note on sanitizing streams with differential privacy. *arXiv preprint*, 2021. arXiv:2111.13762.
- 20 Zohar Karnin, Kevin Lang, and Edo Liberty. Optimal quantile approximation in streams. In *2016 IEEE 57th annual symposium on foundations of computer science (focs)*, pages 71–78. IEEE, 2016.
- 21 Kasper Green Larsen, Rasmus Pagh, and Jakub Tětek. Countsketches, feature hashing and the median of three. In *International Conference on Machine Learning*, pages 6011–6020. PMLR, 2021.
- 22 Christian Janos Lebeda and Jakub Tětek. Better differentially private approximate histograms and heavy hitters using the misra-gries sketch. *arXiv preprint*, 2023. arXiv:2301.02457.
- 23 Alexander J McNeil, Rüdiger Frey, and Paul Embrechts. *Quantitative risk management: concepts, techniques and tools-revised edition*. Princeton university press, 2015.

- 24 Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. *arXiv preprint*, 2015. [arXiv:1508.06110](#).
- 25 Darakhshan Mir, Shan Muthukrishnan, Aleksandar Nikolov, and Rebecca N Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 37–48, 2011.
- 26 Jayadev Misra and David Gries. Finding repeated elements. *Science of computer programming*, 2(2):143–152, 1982.
- 27 Rasmus Pagh and Mikkel Thorup. Improved utility analysis of private counts sketch. *arXiv preprint*, 2022. [arXiv:2205.08397](#).
- 28 Ryan M Rogers, Aaron Roth, Jonathan Ullman, and Salil Vadhan. Privacy odometers and filters: Pay-as-you-go composition. *Advances in Neural Information Processing Systems*, 29, 2016.
- 29 Harry Sivasubramaniam, Haonan Li, and Xi He. Differentially private sublinear average degree approximation, 2020.
- 30 Adam Smith, Shuang Song, and Abhradeep Guha Thakurta. The flajolet-martin sketch itself preserves differential privacy: Private counting with minimal space. *Advances in Neural Information Processing Systems*, 33:19561–19572, 2020.
- 31 Nina Mesing Stausholm. Improved differentially private euclidean distance approximation. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 42–56, 2021.
- 32 Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.
- 33 Lun Wang, Iosif Pinelis, and Dawn Song. Differentially private fractional frequency moments estimation with polylogarithmic space. *arXiv preprint*, 2021. [arXiv:2105.12363](#).
- 34 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum  $\tilde{}$  matchings. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 225–234, 2009.
- 35 Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. Differentially private linear sketches: Efficient implementations and applications. *arXiv preprint*, 2022. [arXiv:2205.09873](#).

# Improved Bounds for Graph Distances in Scale Free Percolation and Related Models

Kostas Lakis ✉ 

ETH Zürich, Department of Computer Science, Zürich, Switzerland

Johannes Lengler ✉

ETH Zürich, Department of Computer Science, Zürich, Switzerland

Kalina Petrova ✉ 

ETH Zürich, Department of Computer Science, Zürich, Switzerland

Leon Schiller ✉

ETH Zürich, Department of Computer Science, Zürich, Switzerland

---

## Abstract

In this paper, we study graph distances in the geometric random graph models scale-free percolation SFP, geometric inhomogeneous random graphs GIRG, and hyperbolic random graphs HRG. Despite the wide success of the models, the parameter regime in which graph distances are polylogarithmic is poorly understood. We provide new and improved lower bounds. In a certain portion of the parameter regime, those match the known upper bounds.

Compared to the best previous lower bounds by Hao and Heydenreich [19], our result has several advantages: it gives matching bounds for a larger range of parameters, thus settling the question for a larger portion of the parameter space. It strictly improves the lower bounds of [19] for all parameters settings in which those bounds were not tight. It gives tail bounds on the probability of having short paths, which imply shape theorems for the  $k$ -neighbourhood of a vertex whenever our lower bounds are tight, and tight bounds for the size of this  $k$ -neighbourhood. And last but not least, our proof is much simpler and not much longer than two pages, and we demonstrate that it generalizes well by showing that the same technique also works for first passage percolation.

**2012 ACM Subject Classification** Mathematics of computing → Random graphs; Mathematics of computing → Stochastic processes; Theory of computation → Random network models

**Keywords and phrases** Mathematics, Probability Theory, Combinatorics, Random Graphs, Random Metric Spaces

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.74

**Category** RANDOM

**Related Version** *Full Version*: <https://arxiv.org/abs/2405.07217>

**Funding** *Kostas Lakis*: gratefully acknowledges support from the John S. Latsis Public Benefit Foundation and the Onassis Foundation

*Kalina Petrova*: funded by SNSF grant No. CRSII5 173721

*Leon Schiller*: funded by SNSF grant No. 197138

**Acknowledgements** This research started at the joint workshop of the *Combinatorial Structures and Algorithms* and *Theory of Combinatorial Algorithms* groups of ETH Zürich held in Stels, Switzerland, January 2024. We thank the organizers for providing a very pleasant and inspiring working atmosphere.

## 1 Introduction and Main results

In the last years, a family of random graph models including *hyperbolic random graphs* (HRG) [26], *scale-free percolation* (SFP) [14], and *geometric inhomogeneous random graphs* (GIRG) [11], has emerged as a model for large real-world networks. They combine an



© Kostas Lakis, Johannes Lengler, Kalina Petrova, and Leon Schiller;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 74; pp. 74:1–74:22



Leibniz International Proceedings in Informatics

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

underlying geometric space with an inhomogeneous degree distribution. This combination yields many properties that occur in real-world networks across a wide range of domains, such as strong clustering, a rich community structure, ultra-small distance, small separators, compressibility, and more [11]. The networks have been used empirically and theoretically to study algorithms like local routing protocols [7, 9, 12], bidirectional search [6] or maximum flow [8], spreading processes like bootstrap percolation [21] or SI models [22–25], and they have been used to study the effectiveness of different interventions during the Covid19 pandemic [17, 20, 27].

Despite this widespread adoption, the fundamental question of graph distances has been open for some parameter regimes. In general, the models come with two parameters: the degrees follow a power-law distribution with exponent  $\tau > 1$ , i.e.,  $\mathbb{P}(\deg(v) \geq x) \sim x^{1-\tau}$  for any fixed vertex  $v$ ;<sup>1</sup> and  $\alpha > 1$  determines the number of *weak ties* [18] in the network, i.e. edges which are present although their geometric distance and degrees suggest otherwise. The parameter  $\alpha$  is also called *inverse temperature*. If  $\tau < 3$  then for two random vertices  $x, y$  in the giant component, with high probability<sup>2</sup> their graph distance  $d_G(x, y)$  is at most doubly logarithmic in their geometric distance, i.e.,  $d_G(x, y) = O(\log \log |x - y|)$ . For  $\tau \leq 2$  we even have  $d_G(x, y) = O(1)$ . These regimes are very precisely understood [1, 10, 14]. On the other hand, if  $\tau > 3$  and  $\alpha > 2$ , then it is known that the graph distance of two vertices  $x, y$  grows linearly with their geometric distance, and this is again well understood [2, 15].

However, in the *polylogarithmic regime*  $\tau > 3$  and  $\alpha \in (1, 2)$ , the picture is incomplete. It is understood in the limiting case  $\tau = \infty$ , which is known as *long-range percolation* LRP, that  $d_G(x, y) = (\log |x - y|)^{\Delta(\alpha) \pm o(1)}$  where  $\Delta(\alpha) = 1/\log_2(2/\alpha) > 1$  [2, 3]. Since graph distances can only increase with  $\tau$ , the upper bound applies for any  $\tau > 1$ , and this is the best known upper bound.<sup>3</sup> On the other hand, it is easy to see that for  $\tau > 3$  the  $k$ -neighbourhood can grow at most exponentially, so distances are at least logarithmic,  $d_G(x, y) = \Omega(\log |x - y|)$  [14]. Hence we know that distances are polylogarithmic for  $\tau > 3$  and  $\alpha \in (1, 2)$ . But the exponents of the upper and lower bound ( $\Delta$  and 1 respectively) did not match, and this gap remained open for a long time.

Very recently, Hao and Heydenreich [19] could show an improved lower bound of  $d_G(x, y) \geq (\log |x - y|)^{\Delta(\min\{\alpha, (\tau-1)/2\}) - o(1)}$ , where, as before,  $\Delta(x) = 1/\log_2(2/x)$ . This closed the gap in the case that  $\alpha < (\tau - 1)/2$ , since then  $\min\{\alpha, (\tau - 1)/2\} = \alpha$ . Their proof had 9 pages and was a complicated application of Biskup’s hierarchy argument [3]. In this paper we give a stronger lower bound with a much simpler inductive proof of only about two pages. It is inspired by ideas of Biskup for the simpler case of LRP [4], which themselves are adaptations of those in [29]. More precisely, we show that  $d_G(x, y) = \Omega((\log |x - y|)^{\Delta(\min\{\alpha, \tau-2-o(1)\})})$ . This closes the gap between upper and lower bound whenever  $\alpha < \tau - 2$ , which comprises a strictly larger portion of the polylogarithmic regime than the bound of [19]. Moreover, throughout the polylogarithmic regime, since  $\tau > 3$  implies  $\tau - 2 > (\tau - 1)/2$ , our bound is strictly stronger in all cases in which the bound of [19] is not tight.

<sup>1</sup> The distribution is often allowed to vary by constant factors or slowly varying functions, but this will not be relevant for this paper. Also, the traditional SFP parameterization uses two parameters  $\tau$  and  $\gamma$  instead of the one parameter  $\tau$ . However, one of those parameters is internal to the graph generation process and does not yield additional classes of graphs, which is why we omit the additional parameter.

<sup>2</sup> We say an event occurs *with high probability*, or w.h.p., if it occurs with probability  $1 - o(1)$ .

<sup>3</sup> An improved upper bound was claimed in [19], but the proof had an issue which we consider severe, see Appendix A for details. At submission time of the camera-ready version of this paper, the problem has not been fixed, so currently we must consider this result as unproven.



Our result is also stronger in two other aspects: Firstly, we provide strong tail bounds on the probability  $\mathbb{P}(d_G(x, y) \leq k)$ . These yield a *shape theorem* for the geometric shape of the  $k$ -neighbourhood of a fixed vertex for growing  $k$ , in the case  $\alpha < \tau - 2$  when our lower bound matches the upper bound. The shape theorem also implies that the size of the  $k$ -neighbourhood of a fixed vertex grows as  $e^{k^{1/\Delta \pm o(1)}}$  as  $k \rightarrow \infty$ , which was not known before. Secondly, due to its simplicity, we believe that our method is also potentially easier to generalize. As demonstration, we show that a similar lower bound holds not only for graph distances, but also for first passage percolation.

In the following, we will start by formally defining the graph models and stating our precise results. The heart of the paper is Section 2, where we prove our result for the SFP model. In Section 3, we extend the proof to first-passage percolation. Finally, in Appendix A we explain why the proof of the upper bound claimed in [19] is incorrect.

## 1.1 Preliminaries and Random Graph Models

Our results hold for *Scale Free Percolation* (SFP), *Geometric Inhomogeneous Random Graphs* (GIRG), and *Hyperbolic Random Graphs* (HRG). We will define LRP and GIRG formally in this section. HRG has been shown to be a special case of GIRG [11], so all results proven for GIRG automatically also hold for HRG, and we do not need to formally define HRG. A formal definition of HRG together with its connection to GIRG can be found in [11].

### 1.1.1 Scale Free Percolation (SFP)

For SFP, we start with the infinite<sup>4</sup>  $d$ -dimensional grid, which is our set of vertices. For two points  $x, y \in \mathbb{Z}^d$ , we define their distance  $|x - y|$  via the usual Euclidean norm. Moreover, each vertex draws a *weight*  $w_x$  independently identically distributed from a power-law distribution. For our purposes, this is a Pareto distribution satisfying

$$\mathbb{P}(w_x \geq z) = z^{1-\tau}$$

for  $z \geq 1$ , where the parameter  $\tau > 1$  is the *power-law exponent*.

We add edges in two different ways. First, we place the usual *grid edges*<sup>5</sup> between points that are adjacent in  $\mathbb{Z}^d$ , i.e., we place an edge between  $x = (x_1, \dots, x_d)$  and  $y = (y_1, \dots, y_d)$  if there is some coordinate  $1 \leq i \leq d$  such that  $|x_i - y_i| = 1$  and  $x_j = y_j$  for all  $j \neq i$ . Second, we randomly create long-range edges, also called *weak ties*, by placing an edge between  $x, y \in \mathbb{Z}^d$ , independently for different sets  $\{x, y\}$ , with probability  $p_{xy}$ , which is defined as<sup>6</sup>

$$p_{xy}^{(\text{SFP})} := \min \left\{ 1, \lambda \left( \frac{w_x w_y}{|x - y|^d} \right)^\alpha \right\}, \quad (1)$$

where  $\lambda > 0, \alpha > 1$  are constants. We write  $x \sim y$  if there is an edge between  $x, y \in \mathbb{Z}^d$  and  $x \approx y$  otherwise.

<sup>4</sup> Traditionally, LRP is defined on an infinite vertex set while GIRG is defined on a finite vertex set, following the tradition of mathematics for LRP and of computer science for GIRG. However, both models can be defined in either a finite or an infinite version, and this does not affect graph distances, see [25] for details.

<sup>5</sup> Some variants of SFP do not include grid edges. Since we prove lower bounds on graph distances, we make our result stronger by including grid edges.

<sup>6</sup> In the literature, the connection probability is often defined as  $1 - \exp(-\lambda|x - y|^{-\alpha d})$ . We remark that this differs from our connection probability at most by a constant factor, which does not change the model substantially.

We remark that the parameterization is slightly different from the original one used in [14] and also in [19]. Compared to our formulation, they used auxiliary weights  $w'_x := w_x^\alpha$ , which were then drawn from a power-law distribution with exponent  $\tau' = 1 + d(\tau - 1)/\alpha$ . These two parameterizations are equivalent, but our formulation has the advantage that the weights correspond, up to constant factors, to the expected degree of the vertices,  $\mathbb{E}[\deg(x) \mid w_x] = \Theta(w_x)$ . Our formulation also saves an internal parameter of the model. Moreover, we rescaled  $\alpha$  by a factor  $d$  to match it with the parameterization of GIRG. When comparing our results with the lower bounds in [19], the following transformations are needed, where the subscript “[19]” indicates notation from that paper, and parameters without subscript are from our paper.

$$\alpha_{[19]} = \alpha d \quad \text{and} \quad \gamma_{[19]} = \tau - 1. \tag{2}$$

The internal parameter  $\tau_{[19]}$  is superfluous and does not have a correspondence in our paper.

### 1.1.2 Geometric Inhomogeneous Random Graphs (GIRG)

The GIRG model is sometimes also referred to as Continuum Scale Free Percolation [16]. The main difference between SFP and GIRG is that in GIRG, the positions of vertices are randomly chosen. Namely, for some large enough  $n \in \mathbb{N}$ , we consider a cube  $\mathcal{X}$  of volume  $n$  in  $\mathbb{R}^d$ , where  $d$  is a constant. Our vertex set  $V$  then consists of  $n$  vertices, where the position  $\xi_x$  of each vertex  $x$  is picked independently at random from the uniform distribution over  $\mathcal{X}$ . The distance between two vertices  $x, y \in V$  is defined as the Euclidean norm  $|\xi_x - \xi_y|$ , as in SFP. Again similarly to SFP, each vertex  $x$  draws a weight  $w_x$  independently from the Pareto distribution satisfying

$$\mathbb{P}(w_x \geq z) = z^{1-\tau}$$

for  $z \geq 1$ , with  $\tau > 1$ . Finally, two different vertices  $x$  and  $y$  are connected by an edge with probability<sup>7</sup>

$$p_{xy}^{(\text{GIRG})} = \Theta \left( \min \left\{ 1, \left( \frac{w_x w_y}{|\xi_x - \xi_y|^d} \right)^\alpha \right\} \right), \tag{3}$$

where the hidden constants are uniform over all  $x, y$ . Formally, we require that there are two absolute constants  $c_{\text{low}}, c_{\text{upp}} > 0$  independent of  $n$  such that for all  $n$  and any two different vertices  $x, y \in V$ , conditional on their weights  $w_x, w_y \geq 1$  and positions  $\xi_x, \xi_y \in \mathcal{X}$ ,

$$c_{\text{low}} \min \left\{ 1, \left( \frac{w_x w_y}{|\xi_x - \xi_y|^d} \right)^\alpha \right\} \leq p_{xy}^{(\text{GIRG})} \leq c_{\text{upp}} \min \left\{ 1, \left( \frac{w_x w_y}{|\xi_x - \xi_y|^d} \right)^\alpha \right\}.$$

The reason for allowing constant factor deviations is that then hyperbolic random graphs (HRG) is a special case of GIRG with  $d = 1$ , where the Euclidean distance is replaced by the angular distance in hyperbolic space [11]. Hence, any statement proven for this version of GIRG also holds for HRG.

Note that the constants  $\tau$  and  $\alpha$  have an analogous role here as in SFP. A notable difference to SFP is that in GIRG there are no grid edges (since positions are no longer on the grid), but that does not significantly affect our results.

---

<sup>7</sup> The original paper [11] used a geometry that was rescaled by a factor  $n^{1/d}$ . I.e., they used a cube of volume one and had an additional factor  $n$  in the denominator of (3). Both variants are equivalent, but our scaling aligns better with the SFP scaling and allows GIRGs to be extended to infinite graphs if desired [25].

### 1.1.3 Long Range Percolation (LRP)

To put our results in an adequate context, we sometimes reference the model of *Long Range Percolation* (LRP) which is a special case and predecessor of SFP, cf. e.g. [3, 5]. We define LRP completely analogously to SFP with the only modification that every vertex has a (deterministic) weight of 1.

### 1.1.4 First Passage Percolation (FPP)

Our lower bounds on graph distances are (in a similar form) also applicable to *First Passage Percolation* (FPP) on the graph models SFP, GIRG, HRG, and also LRP. Here, each edge  $e$  of the graph draws a random length or cost  $c_e$  independently identically distributed (i.i.d.) from a distribution over the non-negative reals. For two vertices  $x, y$  we are then interested in the *minimal/infimal cost* of all paths from  $x$  to  $y$ . This can be done on an arbitrary finite or infinite underlying graph. In this work we restrict our attention to the case where each edge cost is sampled from an exponential distribution with rate 1. The formal definition is as follows.

► **Definition 1** (First Passage Percolation (FPP)). We call the following process *First Passage Percolation*, in short FPP. Given a graph  $G = (V, E)$ , assign to each edge  $e$  an i.i.d. cost  $c_e$  sampled from an exponential distribution with rate 1. For a finite path  $\pi$ , we define the *cost* of that path as

$$c(\pi) = \sum_{e \in \pi} c_e$$

Then, the *cost-distance* or *first passage time* between two vertices  $x$  and  $y$  is the minimum (or infimum) cost of any finite path connecting  $x$  and  $y$ , i.e.

$$d_G^{\text{cost}}(x, y) := \inf\{c(\pi) : \pi \in \mathcal{P}_{x,y}\} \quad \text{for } x, y \in V,$$

where  $\mathcal{P}_{x,y}$  is the set of all finite paths between  $x$  and  $y$ .

Recently, FPP on SFP, GIRG, and HRG was studied [13, 23–25]. In particular, it was shown in [23, 25] that FPP on those graphs exhibits an *explosive* behaviour if the vertex weights have infinite variance (i.e. if  $\tau < 3$ ). This means that the cost-distance between two vertices  $x, y$  converges in distribution against a random variable that is finite almost surely. In the infinite model SFP, this means that there are infinitely many vertices reachable within finite cost from a given vertex; in the finite models GIRG and HRG, a constant fraction of all vertices have cost-distance  $\mathcal{O}(1)$ . We emphasize that this is not true for *graph distances* for  $\tau \in (2, 3)$  since then degrees are finite almost surely, so the number of vertices in graph distance  $C$  is finite/constant for any constant  $C > 0$ .

We remark that a model with a similar name, *long-range first passage percolation* was introduced and studied in [13]. This is *not* FPP on LRP, but a different model with the complete graph on  $\mathbb{Z}^d$  (all edges are present, degrees are infinite), where transmission times are penalized for edges between vertices of large Euclidean distance. Despite the differences, the models are related, and our analysis of FPP uses a coupling to a similar model and is inspired by [13].

### 1.1.5 Terminology

We are interested in the typical *graph distance*  $d_G(x, y)$  between two vertices  $x, y$  in SFP, GIRG, and HRG, and the *cost-distance*  $d_G^{\text{cost}}(x, y)$  in FPP.<sup>8</sup> We are interested in the asymptotic behavior of  $d_G(x, y)$  and  $d_G^{\text{cost}}(x, y)$  in terms of the Euclidean distance  $|x - y|$ . That is, we study how  $d_G(x, y)$  and  $d_G^{\text{cost}}(x, y)$  scale as functions of  $|x - y|$  when  $|x - y| \rightarrow \infty$ . We define the function  $\Delta(\beta) := \frac{1}{\log_2(2/\beta)}$ , which will appear in the exponent governing the polylogarithmic behavior of graph distances. Throughout, we use  $\Delta = \Delta(\alpha)$  where  $\alpha$  is the long-range parameter of the relevant model.

Previous work showed that both LRP and SFP exhibit multiple phase transitions in the asymptotic behavior of  $d_G(x, y)$  depending on the model parameters, which were briefly summarized in the introduction and are further summarized (together with our results) in Table 1.

■ **Table 1** Upper and lower bounds for graph distances and cost-distances in long-range percolation (LRP), scale-free percolation (SFP) and first passage percolation (FPP) on SFP, with  $\Delta(\beta) = 1/\log_2(2/\beta)$ . The results on graph distance for SFP also hold for geometric inhomogeneous random graphs (GIRG) and hyperbolic random graphs (HRG). Our results are indicated in bold.

| Model      | $\alpha \in (1, 2)$                             |  | $\alpha > 2$                      |                       |
|------------|---|--|-----------------------------------|-----------------------|
| LRP        | $\Theta(\log( x - y )^{\Delta(\alpha)})$ [3, 5] |  | $\Theta( x - y )$ [2]             |                       |
| SFP        | $\tau \in (2, 3)$                               | $\tau > 3$   | $\tau \in (2, 3)$                 | $\tau > 3$            |
|            | $\Theta(\log \log( x - y ))$ [14]               | $\leq \log( x - y )^{\Delta(\alpha)+o(1)}$ [3, 5] <sup>9</sup><br>$\geq \log( x - y )^{\Delta(\min\{\alpha, \tau-2\})-o(1)}$<br><b>Corollary 2</b> | $\Theta(\log \log( x - y ))$ [14] | $\Theta( x - y )$ [2] |
| FPP on SFP | $\Theta(1)$ [23]                                | $\leq \log( x - y )^{\Delta(\alpha)+o(1)}$ [23]<br>$\geq \log( x - y )^{\Delta(\min\{\alpha, \frac{\tau-1}{2}\})-o(1)}$<br><b>Corollary 3</b>      | $\Theta(1)$ [23]                  | ?                     |

## 1.2 Our results

In this section, we formally state our main results. To keep the exposition simple, we state them only for SFP, not for GIRG and HRG. All results on graph distances in this section also hold for GIRG and HRG, where all constants can be chosen independently of the number  $n$  of vertices. We give details on that in the full version but omit them here due to space constraints.

We prove stronger lower bounds for graph distances in the logarithmic regimes of SFP with a much simpler proof than in [19]. Furthermore, we also obtain a *shape theorem* which sandwiches the  $k$ -neighbourhood of a given vertex between two geometric balls of similar size. The key result is a general upper bound on the probability that two vertices have graph/cost-distance at most  $k$ . For the following claims, recall that  $\Delta(\beta) = 1/\log_2(2/\beta)$ .

► **Theorem 1.1** (Tail Bound for Graph Distances in SFP). Consider SFP with parameters  $\alpha \in (1, 2), \tau > 3$  and  $\lambda$ . Fix any sufficiently small  $\varepsilon > 0$  and let  $\Delta' = \Delta(\min\{\alpha, \tau - 2 - \varepsilon\})$ . Then, there exist constants  $c_1, c_2, \beta$  depending on the model parameters as well as  $\varepsilon$  such that for any pair of vertices  $x, y \in \mathbb{Z}^d$  and any  $k \in \mathbb{N}$ , we have

$$\mathbb{P}(d_G(x, y) \leq k) \leq c_2^{-1} |x - y|^{-\alpha d} (k + 1)^{-\beta} \exp\left(c_1 k^{1/\Delta'}\right).$$

<sup>8</sup> Formally, the graph distance is also defined in Definition 1 by setting  $c(e) := 1$  for all edges  $e$ .

<sup>9</sup> In [19], the authors claim to prove an improved logarithmic upper bound with exponent  $\Delta(\min\{\alpha, \tau - 2\})$ , which would match our lower bounds. However, we show that their proof is wrong, see Appendix A.

The above bound is proved by induction over  $k$  in Section 2. We express the probability that a path of length  $\leq k$  exists recursively by decomposing the path into two parts connected by an edge which has the longest geometric distance on the original path. Applying the inductive hypothesis and integrating over all possible endpoints and weights of the endpoints of said edge then yields the desired bound. However, this actually only works if  $\alpha < \tau - 2$ , which is needed to ensure convergence of some involved integrals. We remedy this and make the theorem applicable also to the case  $\alpha \geq \tau - 2$  by using a coupling argument. Here (see Lemma 5 for the exact statement), we argue that decreasing  $\alpha$  only makes the model denser, so graph distances can only become smaller. Thus, to prove the claim for some  $\alpha \geq \tau - 2$ , we can decrease  $\alpha$  to some value  $\alpha' := \tau - 2 - \varepsilon$  without increasing distances, and then apply Theorem 1.1 for the already settled case  $\alpha'$ . This is the reason why we use the exponent  $\Delta(\min\{\alpha, \tau - 2 - \varepsilon\})$ . Using the above tail bound then directly implies a bound on typical graph distances.

► **Corollary 2** (Typical Graph Distances in SFP). *Consider SFP under the assumption  $\alpha \in (1, 2)$  and  $\tau > 3$ . Then for every sufficiently small  $\varepsilon > 0$ , there is a constant  $c > 0$  such that*

$$\lim_{|x-y| \rightarrow \infty} \mathbb{P} \left( d_G(x, y) \geq c \log(|x - y|)^{\Delta(\min\{\alpha, \tau - 2 - \varepsilon\})} \right) = 1.$$

Note that in the case  $\alpha < \tau - 2$ , the exponent is exactly  $\Delta(\alpha)$  (if  $\varepsilon$  is chosen sufficiently small), which is a slightly stronger result than the one we obtain if  $\alpha \geq \tau - 2$  and matches the known upper bounds in [5] up to only a constant factor in front of the log. The previously best lower bounds by Hao and Heydenreich [19] only matched these upper bounds if  $2\alpha < \tau - 2$  and nonetheless were only tight if we ignore an additional additive constant of  $-\varepsilon$  in the exponent. If  $\alpha \geq \tau - 2$ , we also have to account for such an  $\varepsilon$  in the exponent, but even in this case, our result strengthens the lower bounds in [19] and at the same time relies on a significantly simpler proof. The original proof heavily relied on so-called *hierarchies* as introduced in [3] and required complex combinatorial estimates for showing that certain structures w.h.p. do not exist. We avoid this by using the inductive proof strategy as described above instead. This not only simplifies and improves the existing lower bounds on graph distances, but the tail bound in Theorem 1.1 further yields a so-called *shape theorem* precisely characterizing the diameter and the cardinality of the  $k$ -neighbourhood of a vertex  $x$ , i.e., the set of vertices in graph distance at most  $k$  from  $x$ . To this end, we define  $\mathcal{B}(x, k) := \{y \in V \mid d_G(x, y) \leq k\}$  as the set of all  $k$ -hop neighbors of a given vertex  $x$ .

► **Theorem 1.2** (Shape Theorem for  $k$ -Balls in SFP). *Consider SFP with  $\alpha < \tau - 2$ . Let  $\Delta = \Delta(\alpha)$ , fix an  $\varepsilon > 0$  and let  $\mathcal{X}_{low}, \mathcal{X}_{upp}$  be the set of vertices at a geometric distance of at most  $q(k) = e^{k^{1/\Delta - \varepsilon}}$  and at most  $r(k) = e^{k^{1/\Delta + \varepsilon}}$  from a fixed vertex  $x$ , respectively. Then, we have*

$$\lim_{k \rightarrow \infty} \mathbb{P}(\mathcal{X}_{low} \subseteq \mathcal{B}(x, k) \subseteq \mathcal{X}_{upp}) = 1. \quad (\text{A})$$

In particular,

$$\lim_{k \rightarrow \infty} \mathbb{P} \left( e^{k^{1/\Delta - \varepsilon}} \leq |\mathcal{B}(x, k)| \leq e^{k^{1/\Delta + \varepsilon}} \right) = 1. \quad (\text{B})$$

The lower bound of the theorem comes from [4], while we contribute the upper bound, which is a relatively straightforward corollary of our tail bound in Theorem 1.1. For this reason, we defer the formal proof to the full version of this paper. Note that lower bounds on graph distances correspond to upper bounds for  $\mathcal{B}(x, k)$  and vice versa.

We further note that while our upper bound in Theorem 1.2 also holds for GIRG and HRG, the lower bound is not known for these models and can only hold with some caveats. Firstly, due to the lacking grid edges those graphs are not connected, and require additional constraints to ensure the existence of a giant (linear-size) connected component. Second, even if the giant component exists, a constant fraction of vertices are not in the largest component. Hence, the lower bounds in (A) and (B) can only hold conditioned on  $x$  being in the largest component, and for (A) we must intersect  $\mathcal{X}_{low}$  with the giant component. We conjecture that the lower bounds in Theorem 1.2 hold with these caveats, but this is not known.

### 1.3 First passage percolation on SFP

Using similar techniques and inspired from those in [13], we can prove similar statements for FPP on SFP. Analogous to Theorem 1.1, we obtain a tail bound on the probability that  $x, y$  have cost-distance at most  $t$ . We remark that the same result could be obtained analogously for GIRG (and thus HRG) as well, but we omit this for conciseness.

► **Theorem 1.3** (Tail Bound for Cost-Distances for FPP on SFP). Consider FPP on SFP and arbitrary vertices  $x, y$ . Fix any sufficiently small  $\varepsilon > 0$ . There exists a constant  $c$  depending only on  $\alpha, \varepsilon$  and  $\tau$  such that for  $\Delta'' = \Delta(\min\{\alpha, \frac{\tau-1}{2}\} - \varepsilon)$ ,

$$\mathbb{P}(d_G^{\text{cost}}(x, y) \leq t) \leq |x - y|^{-\alpha d} \exp\left(ct^{1/\Delta''}\right).$$

The proof of this differs from Theorem 1.1 in some significant aspects which are formally presented in Section 3. Intuitively, the main differences are as follows. Firstly, we now have two sources of randomness: the existence of edges and the cost of an existing edge. The first step towards proving Theorem 1.3 is therefore to combine these two sources into a single one. This is achieved by coupling the model to a related model called *Complete Scale Free First Passage Percolation* or CFFP for short. Here, all possible edges on the vertex set  $\mathbb{Z}^d$  exist a priori but the cost of the edge between  $x, y \in \mathbb{Z}^d$  is now drawn from an exponential distribution with rate  $w_x^\alpha w_y^\alpha |x - y|^{-\alpha d}$  instead of rate 1, i.e., in CFFP the rate of an edge depends on the vertex weights and (geometric) distances of its endpoints.

The second main difference to the proof of Theorem 1.1 is that cost-distances are continuous random variables, so we cannot union-bound over all possible cost-distances before and after the longest edge of a potential path anymore like we did for SFP (in Lemma 4). Instead, we establish a continuous analog, a so called *self-bounding inequality* that relates the expected size of a  $k$ -ball to itself recursively. Another difficulty one has to overcome is that, in principle, paths of low cost-distance do not necessarily have to correspond to low graph distance as well. It could theoretically happen that many edges have very low cost and we get a low cost path which uses many edges. In such cases, we cannot use the existence of a geometrically long edge in the path, which is very central to our proof for graph distances. However, we are able to show that paths with high graph distance are actually very unlikely to have low cost-distance (see Lemma 8). Finally, a further obstacle in adapting the proof is that we have to work with the probability that a path of a certain cost exists conditioned on the weights of its endpoints at multiple points. This impacts the probabilities of edges/paths existing and thus introduces complications. To overcome this, we relate said probabilities conditional on the involved weights to their unconditional versions by employing a coupling (Proposition 12).

As a corollary of the above tail bound, we obtain lower bounds on the typical cost-distance similar to the one established for SFP (Corollary 2).

► **Corollary 3** (Typical Graph Distances in FPP on SFP). *Consider FPP on SFP under the assumption  $\alpha \in (1, 2)$  and  $\tau > 3$ . Then, for every sufficiently small  $\varepsilon > 0$*

$$\lim_{|x-y| \rightarrow \infty} \mathbb{P} \left( d_G^{\text{cost}}(x, y) \geq \log(|x-y|)^{\Delta(\min\{\alpha, \frac{\tau-1}{2}\}-\varepsilon)} \right) = 1.$$

### 1.3.1 Asymptotics and Probability Theory

We use standard Landau notation for indicating the asymptotic growth of a function. All asymptotic statements refer to the asymptotic behavior of a function as the distance  $|x-y|$  tends to infinity, unless explicitly noted otherwise (like for the shape theorem, where we consider  $k \rightarrow \infty$ .) We further require a version of the Van den Berg-Kesten inequality (or BK-inequality) from [28], which allows us to bound the probability that there exist *disjoint* subpaths connecting a vertex  $x$  to  $u$  and a vertex  $v$  to  $y$  by the product of the probabilities of either path existing, as if they were independent. We refer the interested reader to the full version of this paper for further details.

## 2 Lower Bounds for Graph Distances in SFP

In this section, we provide the proof of our main lower bound. Our proof generally follows the structure of the proof of Theorem 3.1 in [4]. Our goal is to show that the logarithmic exponent in the distances is at least roughly  $\Delta(\min\{\alpha, \tau - 2\})$ . When  $\alpha < \tau - 2$ , this is the same as  $\Delta = \Delta(\alpha)$ . We will first give the proof under this condition, so we first show Lemma 4.

► **Lemma 4** (Tail Bound for Graph Distances in SFP). *Consider SFP with  $\alpha \in (1, 2)$  such that  $\alpha < \tau - 2$  and  $\Delta = \Delta(\alpha)$ . There exist constants  $c_1, c_2, \beta$  depending only on the model parameters, such that for any pair  $x, y \in \mathbb{Z}^d$  and any  $k \in \mathbb{N}$ ,*

$$\mathbb{P}(d_G(x, y) \leq k \mid w_x, w_y) \leq w_x^\alpha w_y^\alpha c_2^{-1} |x-y|^{-\alpha d} (k+1)^{-\beta} e^{c_1 k^{1/\Delta}}. \quad (4)$$

**Proof.** First of all, note that for fixed  $\beta, c_2$ , and  $\lambda$ , the base case ( $k = 1$ ) is true for  $c_1$  large enough. That is because the RHS of 4 is  $2^{-\beta} e^{c_1} w_x^\alpha w_y^\alpha c_2^{-1} |x-y|^{-\alpha d}$  and the actual connection probability is at most  $\lambda w_x^\alpha w_y^\alpha |x-y|^{-\alpha d}$ . For the inductive step, let  $h$  be the RHS of our induction hypothesis, i.e.,

$$h(r, k, w_x, w_y) := w_x^\alpha w_y^\alpha c_2^{-1} r^{-\alpha d} (k+1)^{-\beta} e^{c_1 k^{1/\Delta}}.$$

Assume that the induction hypothesis is true up to  $k-1$ . For  $x, y$  to be connected with at most  $k$  steps, an edge must be used with geometric distance at least  $\frac{|x-y|}{k}$ . This could either be the first or last edge on the path, or a so-called *internal* edge. Let us first bound the probability corresponding to this edge being internal. For this, we union bound over all possible endpoints  $u$  and  $v$  of said edge. Actually, we integrate, since constant factors are essentially immaterial for the proof. At a given distance  $r$ , there are at most  $c_d r^{d-1}$  vertices, for some constant  $c_d$ . Let  $w_u, w_v$  be the weights of  $u, v$ , respectively. By the BK inequality, we have

$$\begin{aligned} & \mathbb{P}(d_G(x, y) \leq k, \text{longest edge is internal} \mid w_u, w_v, w_x, w_y) \leq \\ & \sum_{i=1}^{k-2} \mathbb{P}(d_G(x, u) \leq i \mid w_x, w_u) \mathbb{P}(u \sim v \mid w_u, w_v) \mathbb{P}(d_G(v, y) \leq k-i \mid w_v, w_y) \leq \lambda w_u^\alpha w_v^\alpha \left(\frac{|x-y|}{k}\right)^{-\alpha d} \\ & \times \underbrace{\sum_{i=1}^{k-2} \left( \int_1^\infty c_d r_u^{d-1} \min\{1, h(r_u, i, w_x, w_u)\} dr_u \right)}_{\mathcal{I}(u, i)} \underbrace{\left( \int_1^\infty c_d r_v^{d-1} \min\{1, h(r_v, k-i, w_v, w_y)\} dr_v \right)}_{\mathcal{I}(v, k-i)}. \end{aligned}$$

We will now argue that there exists a constant  $C_{\text{int}}$  (depending on the model parameters) such that

$$\begin{aligned} \mathcal{I}(u, i) & \leq C_{\text{int}} \left( (i+1)^{-\frac{\beta}{\alpha}} e^{\frac{c_1}{\alpha} i^{1/\Delta}} w_u w_x c_2^{-\frac{1}{\alpha}} \right), \\ \mathcal{I}(v, k-i) & \leq C_{\text{int}} \left( (k-i+1)^{-\frac{\beta}{\alpha}} e^{\frac{c_1}{\alpha} (k-i)^{1/\Delta}} w_v w_y c_2^{-\frac{1}{\alpha}} \right). \end{aligned}$$

To this end, note that there exists a value

$$\hat{r}_u = (w_x w_u)^{\frac{1}{d}} c_2^{-\frac{1}{\alpha d}} (i+1)^{-\frac{\beta}{\alpha d}} e^{\frac{c_1}{\alpha d} (i+1)^{1/\Delta}}$$

for  $r_u$  below which the minimum inside the integral  $\mathcal{I}(u, i)$  is 1. We can thus express  $\mathcal{I}(u, i) = \int_1^{\hat{r}_u} f_1(r_u) dr_u + \int_{\hat{r}_u}^\infty f_2(r_u) dr_u$  where  $f_1(r) = c_d r^{d-1}$  and  $f_2$  is a polynomial in  $r_u$  with exponent smaller than  $-1$ . Therefore, the entire integral is dominated by the value of the antiderivative of  $f_1$  and  $f_2$  at the splitting point  $\hat{r}_u$ . Since  $f_1, f_2$  are polynomials, the antiderivative of  $f_1$  is  $\leq c r_u f_1(r_u)$  and the antiderivative of  $f_2$  is  $\leq c r_u f_2(r_u)$  for some constant  $c$ . Since the minimum is a continuous function, we have  $f_1(\hat{r}_u) = f_2(\hat{r}_u)$  and thus,  $\mathcal{I}(u, i) = \Theta(\hat{r}_u f_1(\hat{r}_u)) = \Theta(\hat{r}_u^d)$  as claimed. A similar argument holds for  $\mathcal{I}(v, k-i)$ .<sup>10</sup>

Plugging this in, we obtain

$$\begin{aligned} & \mathbb{P}(d_G(x, y) \leq k, \text{longest edge is internal} \mid w_u, w_v, w_x, w_y) \\ & \leq C_{\text{int}}^2 \lambda \cdot w_u^{1+\alpha} w_v^{1+\alpha} w_x^\alpha w_y^\alpha |x-y|^{-\alpha d} c_2^{-\frac{2}{\alpha}} \\ & \quad \times \underbrace{k^{\alpha d} \sum_{i=1}^{k-2} \left( (i+1)^{-\frac{\beta}{\alpha}} e^{\frac{c_1}{\alpha} i^{1/\Delta}} \right) \left( (k-i+1)^{-\frac{\beta}{\alpha}} e^{\frac{c_1}{\alpha} (k-i)^{1/\Delta}} \right)}_{:=S}. \end{aligned}$$

Our goal now is to show that the above term is at most  $h(r, k, w_x, w_y)$ . To this end, we show that

$$S \leq (k+1)^{-\beta} e^{c_1 k^{1/\Delta}} \tag{5}$$

for  $k$  and  $\beta$  large enough. For this, notice that for small or large  $i$ , the exponential terms in  $S$  are still quite ‘‘tame’’, due to the  $\frac{1}{\alpha}$  factor. When  $i$  is around  $\frac{k}{2}$ , their product (which is maximized for such  $i$  due to concavity) is practically

$$\exp\left(\frac{2c_1}{\alpha} \left(\frac{k}{2}\right)^{1/\Delta}\right) = \exp\left(\frac{2c_1}{\alpha} 2^{-1/\Delta} k^{1/\Delta}\right) = \exp\left(c_1 k^{1/\Delta}\right)$$

<sup>10</sup>This observation is helpful whenever we integrate a continuous and piecewise polynomial function.



as  $2^{-1/\Delta} = \alpha/2$  by definition of  $\Delta$ . In fact, for  $i = \frac{k}{2}$ , we have actual equality and for every  $1 \leq i \leq k$ , we have

$$\exp\left(\frac{c_1}{\alpha} i^{1/\Delta}\right) \exp\left(\frac{c_1}{\alpha} (k-i)^{1/\Delta}\right) \leq \exp\left(c_1 k^{1/\Delta}\right).$$

This is precisely the exponential term that appears in the statement we want to prove (5). However, we also need to account for the sum and the terms polynomial in  $k$  that appear in  $S$ . To this end, we use that – if  $i \approx k/2$  – we gain from the product of the polynomial terms in the sum to compensate overheads. On the other hand, if  $i$  is large or small, the product of the exponential terms is much smaller than what we need, so we can compensate the other terms by using the arising gap. With this in mind, we split the sum in  $S$  into the cases where  $|i - \frac{k}{2}| \leq \frac{k}{4}$  and those where this is not true. This way, we obtain

$$S \leq k^{1+\alpha d} e^{(1-\gamma)c_1 k^{1/\Delta}} + k^{1+\alpha d} ((k+1)/8)^{-\frac{2\beta}{\alpha}} e^{c_1 k^{1/\Delta}}, \quad (6)$$

where  $\gamma > 0$  is a constant depending on  $\Delta$  (and therefore on  $\alpha$ ). The first term accounts for cases where  $i$  is sufficiently far from  $\frac{k}{2}$ , making the exponential terms merge in a tame way. When  $i \in [\frac{k}{4}, \frac{3k}{4}]$ , both  $i+1$  and  $k-i+1$  are at least  $\frac{k+1}{8}$ , since  $k > 2$  (recall that we are analyzing the case where  $k$  edges allow for an internal edge), and this is how the other term is obtained.

Now, notice that since  $\alpha < 2$ , we can choose  $\beta$  large enough such that  $(\alpha d + 1) - \frac{2\beta}{\alpha} < -\beta$ . Then, the second term in 6 is at most  $(k+1)^{-\beta} e^{c_1 k^{1/\Delta}}$  as desired. For the first term, we notice that the same holds if  $k$  is large enough. Hence, for all  $k$ ,  $S$  is at most some constant  $C$  times  $(k+1)^{-\beta} e^{c_1 k^{1/\Delta}}$ . We use this to conclude that

$$\begin{aligned} \mathbb{P}(d_G(x, y) \leq k, \text{ longest edge is internal} \mid w_u, w_v, w_x, w_y) \\ \leq C C_{\text{int}}^2 \lambda c_2^{1-\frac{2}{\alpha}} w_u^{\alpha+1} w_v^{\alpha+1} w_x^\alpha w_y^\alpha |x-y|^{-\alpha d} (k+1)^{-\beta} e^{c_1 k^{1/\Delta}} c_2^{-1} \\ = C C_{\text{int}}^2 \lambda c_2^{1-\frac{2}{\alpha}} w_u^{\alpha+1} w_v^{\alpha+1} \cdot h(|x-y|, k, w_x, w_y). \end{aligned}$$

Since we assume that  $\alpha < \tau - 2$ , we can integrate  $w_u, w_v$  out such that the corresponding integrals over  $w_u$  and  $w_v$  converge and only obtain another constant factor overhead. Then, we can choose  $c_2$  large enough to compensate these constant overheads. Notice that this works since we have a factor of  $c_2^{1-\frac{2}{\alpha}}$  where the exponent is negative because  $\alpha < 2$ . In total, we have shown that we can choose the constants  $\beta, c_1$  and  $c_2$  such that the above bound is at most  $\frac{1}{3} h(|x-y|, k, w_x, w_y)$  for all  $k$ .

Now, let us also bound the probability of paths in which the longest edge is adjacent to either  $x$  or  $y$ . To this end, we sum over all possible vertices  $z$  connected to  $x$  by an edge of (geometric) length  $\geq |x-y|/k$ . Again, by the BK inequality, we have

$$\begin{aligned} \mathbb{P}(d_G(x, y) \leq k, \text{ longest edge incident to } x \mid w_x, w_z, w_y) \\ \leq \lambda w_x^\alpha w_z^\alpha \left(\frac{|x-y|}{k}\right)^{-\alpha d} \left(\int_1^\infty c_d r^{d-1} \min\{1, h(r, k-1, w_z, w_y)\} dr\right) \\ \leq C \lambda w_x^\alpha w_z^\alpha \left(\frac{|x-y|}{k}\right)^{-\alpha d} k^{-\frac{\beta}{\alpha}} e^{\frac{c_1}{\alpha} k^{1/\Delta}} w_z w_y c_2^{-\frac{1}{\alpha}} \\ \leq C \lambda w_z^{\alpha+1} \cdot (k+1)^\beta k^{\alpha d - \frac{\beta}{\alpha}} e^{c_1(\frac{1}{\alpha}-1)k^{1/\Delta}} c_2^{1-\frac{1}{\alpha}} \cdot h(|x-y|, k, w_x, w_y). \end{aligned}$$

Again, by integrating out  $w_z$ , we get another constant factor. We can now choose  $c_1$  large enough so that the term exponential in  $k$  (which has a negative exponent since  $\frac{1}{\alpha} - 1 < 0$ ) swallows the polynomial and constant terms for every  $k$ , ensuring that the factor in front

of  $h(|x - y|, k, w_x, w_y)$  is at most  $\frac{1}{3}$ , as desired. Finally, summing the three possibilities that the longest edge is internal, the first, or the last edge on the path yields that overall,  $\mathbb{P}(d_G(x, y) \leq k \mid w_x, w_y) \leq h(|x - y|, k, w_x, w_y)$  and finishes the proof.  $\blacktriangleleft$

As promised, we now deal with cases where  $\alpha \geq \tau - 2$  by coupling SFP to SFP with larger  $\alpha$  without decreasing distances using the following lemma.

**► Lemma 5.** *Let  $\alpha$  and  $\lambda$  be the long-range and percolation parameters of some instance of SFP. Fix the weights of all vertices and let  $p_{uv}$  refer to the probability that two vertices  $u$  and  $v$  are connected by an edge. Fix some  $\alpha' < \alpha$ . Then,  $p_{uv} \leq \min\{1, \lambda^{\alpha'/\alpha} w_u^{\alpha'} w_v^{\alpha'} |u - v|^{-d\alpha'}\}$ . In particular, this means that the original SFP graph (with parameter  $\alpha$ ) is a subgraph of the one with parameters  $\alpha'$  and  $\lambda' = \lambda^{\alpha'/\alpha}$ .*

**Proof.** We have

$$\begin{aligned} p_{uv} &\leq \min\{1, \lambda w_u^\alpha w_v^\alpha |u - v|^{-d\alpha}\} = \left(\min\{1, \lambda^{\frac{1}{\alpha}} w_u w_v |u - v|^{-d}\}\right)^\alpha \\ &\leq \left(\min\{1, \lambda^{\frac{1}{\alpha}} w_u w_v |u - v|^{-d}\}\right)^{\alpha'} = \min\{1, \lambda^{\frac{\alpha'}{\alpha}} w_u^{\alpha'} w_v^{\alpha'} |u - v|^{-d\alpha'}\}. \end{aligned} \quad \blacktriangleleft$$

The implication of Lemma 5 is that we can artificially ensure that  $\alpha < \tau - 2$  by setting  $\alpha' = \tau - 2 - \varepsilon$  for an arbitrarily small  $\varepsilon$  and  $\lambda' = \lambda^{\alpha'/\alpha}$ . This allows us to prove Theorem 1.1 by applying Lemma 4 to this model since here, graph distances only get shorter due to Lemma 5. We defer the proof to the full version, since it is only technical and the ideas in it are already presented.

### 3 First Passage Percolation (FPP)

In this section we study first passage percolation (FPP) on SFP. Recall that this means that we assign a cost to every edge which is drawn independently from an exponential distribution with rate 1. For conciseness, we restrict ourselves to SFP even though the same technique would also work for GIRGs/HRGs. Note that we obtain the LRP model from SFP by informally setting  $\tau = \infty$ . Formally, since SFP is an increasing model in  $\tau$  in terms of stochastic domination, the edge set of SFP with any finite  $\tau$  stochastically dominates the edge set of LRP. Hence, all lower bounds on cost-distances from SFP also transfer to LRP.<sup>11</sup> In the following, we assume for simplicity that  $\lambda = 1$ ; this does not affect our results.

In contrast to plain SFP, in FPP we have an additional source of randomness since not only the existence of an edge is random but also its cost. To prove lower bounds on cost-distances, it is therefore simpler (and sufficient) to consider a model in which there is only one source of randomness for the edges. We call this model *Complete Scale Free First Passage Percolation*, or CFFP for short. Here, all edges exist a priori, i.e., the graph is fixed to be the complete graph with vertex set  $\mathbb{Z}^d$ . However, we now draw the cost of each edge by sampling from an exponential distribution with rate  $w_u^\alpha w_v^\alpha |u - v|^{-\alpha d}$  (i.e. a rate that depends on the weights and geometric distance between the two endpoints) instead of rate 1.

We start by showing that FPP on SFP is dominated by CFFP, i.e., that cost-distances in CFFP can only become shorter as compared to FPP on SFP. To that end, we need the following lemma that will allow us to combine the randomness of two events occurring with probability  $\min\{1, \alpha\}$  and  $(1 - e^{-b})$ , respectively into an event occurring with probability  $1 - e^{-ab}$ .

<sup>11</sup>The same is true for graph-distances, but here the results for LRP were already known.

► **Lemma 6.**  $\min\{1, a\}(1 - e^{-b}) \leq 1 - e^{-ab}$  for all  $a, b \geq 0$ .

**Proof.** If  $a \geq 1$ , then the inequality is easy to see, as  $(1 - e^{-b}) \leq (1 - e^{-ab})$  in this case. So, let us assume that  $a < 1$  from now on. Consider the function

$$f(b) = a(1 - e^{-b}) - (1 - e^{-ab}).$$

Note that it suffices to show that  $f(b) \leq 0$  for all  $b \geq 0$ . We can see that  $f(0) = 0$  and also

$$f'(b) = a(e^{-b} - e^{-ab}) \leq 0.$$

This shows that the function  $f(b)$  is non-increasing and since  $f(0) = 0$ , we have  $f(b) \leq 0$  for all  $b \geq 0$ . ◀

With this, we establish a coupling between FPP on SFP and CFFP such that cost-distances in CFFP are at most as large as cost-distances in FPP on SFP.

► **Lemma 7.** Let  $u, v$  be a pair of vertices in  $\mathbb{Z}^d$ . Let further  $X_{(u,v)}$  be the cost of the edge  $\{u, v\}$  in FPP on SFP if it exists, and  $X_{(u,v)} = \infty$  if the edge does not exist, and let  $Y_{(u,v)}$  be its cost in CFFP. Then for any  $t \geq 0$ ,

$$\mathbb{P}(X_{(u,v)} \leq t) \leq \mathbb{P}(Y_{(u,v)} \leq t).$$

**Proof.** For the event on the LHS to be true, the edge  $\{u, v\}$  must exist and then *independently* the cost must be drawn to be at most  $t$ . The probability for the first event is  $\min\{1, (w_u w_v)^\alpha |u - v|^{-\alpha d}\}$  and the probability of the latter is  $1 - e^{-t}$ . For the event on the RHS, one simply needs that the cost sampled from an exponential distribution with rate  $(w_u w_v)^\alpha |u - v|^{-\alpha d}$  is at most  $t$  and the probability of this is exactly  $1 - e^{-(w_u w_v)^\alpha |u - v|^{-\alpha d} t}$ . Lemma 6 finishes the proof. ◀

Lemma 7 shows that any lower bound shown for cost-distances in CFFP will also be true for FPP on SFP. To see more clearly why this is true, note that we can couple the models in the following way. First, we sample the weights for the vertices in exactly the same way for both models. Then, conditioned on these weights, the probability space is a product space over independent one-dimensional random variables (technically, one of them can be infinite in value, but this is not a problem for our purposes) for which the inequality in Lemma 7 holds. With this in mind, we continue by establishing the lower bound for cost-distances in CFFP. We will generally follow similar arguments as the ones presented in [13], which studies a model similar to CFFP but without vertex weights. To establish an upper bound on the probability that the cost-distance between two vertices is at most  $t$ , we need a bound on the probability that the sum of exponential random variables is at most  $t$ , which is provided in the following lemma, which in turn is an adaptation of Lemma 2.1 in [13].

► **Lemma 8.** Let  $X_1, X_2, \dots, X_k$  be i.i.d. exponential random variables such that the rate of  $X_i$  is  $(w_i w_{i+1})^\alpha |u_i - u_{i+1}|^{-\alpha d}$ , for some sequence of vertices  $u_j$  with corresponding weight  $w_j$ , with  $1 \leq j \leq k + 1$ . The  $w_i$  are drawn from a power law with exponent  $\tau$ . Assume that  $2\alpha < \tau - 1$ . Then, there exists a  $c > 0$  depending only on  $\alpha, \tau$  such that for all  $t \geq 0$ ,

$$\mathbb{P}\left(\sum_{i=1}^k X_i \leq t\right) \leq \left(\frac{ect}{k}\right)^k \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d},$$

where the above probability is taken over the randomness of the weights and the  $X_i$  values.

**Proof.** Note that each  $X_i = \frac{Y_i}{(w_i w_{i+1})^\alpha |u_i - u_{i+1}|^{-\alpha d}}$ , where  $Y_i$  is an exponential random variable with rate 1. Let us use  $\lambda_i = (w_i w_{i+1})^\alpha |u_i - u_{i+1}|^{-\alpha d}$  from now on. By Markov's inequality, we have

$$\mathbb{P}\left(\sum_{i=1}^k X_i \leq t\right) = \mathbb{P}\left(\exp\left(-\theta \sum_{i=1}^k X_i\right) \geq e^{-\theta t}\right) \leq e^{\theta t} \mathbb{E}\left[\exp\left(-\theta \sum_{i=1}^k X_i\right)\right].$$

Now, let us bound the expectation above. Once one fixes the weights  $w_j$ , each  $X_i$  is independent from each other. Moreover, for  $Y_i$  with rate one, it holds that  $\mathbb{E}[\exp(-\theta Y_i)] = \frac{1}{1+\theta} \leq \frac{1}{\theta}$  for  $\theta > 0$ . So, for a fixed realization  $w_1, w_2, \dots, w_{k+1}$  of the weights, we have:

$$\begin{aligned} \mathbb{E}\left[\exp\left(-\theta \sum_{i=1}^k X_i\right) \mid w_1, w_2, \dots, w_{k+1}\right] &= \prod_{i=1}^k \mathbb{E}\left[\exp\left(-\frac{\theta}{\lambda_i} Y_i\right)\right] \\ &\leq \prod_{i=1}^k \frac{\lambda_i}{\theta} \\ &\leq \theta^{-k} \prod_{i=1}^{k+1} (w_i)^{2\alpha} \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d}. \end{aligned}$$

The weight terms are raised to  $2\alpha$ , since each weight  $w_i$  enters in (at most) two  $\lambda_j$  as  $w_i^\alpha$ . Integrating the weights out, we see that since they are independent, one has

$$\mathbb{E}\left[\exp\left(-\theta \sum_{i=1}^k X_i\right)\right] \leq \left[\theta^{-k} \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d}\right] \prod_{i=1}^{k+1} \mathbb{E}[(w_i)^{2\alpha}].$$

Now, since  $2\alpha - \tau < -1$ , the expectations inside the rightmost product are all at most some constant  $c'$ . Let  $c$  be e.g. equal to  $(c')^2$  such that  $c^k \geq (c')^{k+1}$ . Collecting the above bounds, we have

$$\mathbb{P}\left(\sum_{i=1}^k X_i \leq t\right) \leq e^{\theta t} \left[\theta^{-k} \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d}\right] c^k.$$

Setting  $\theta = \frac{k}{t}$  shows the desired bound.  $\blacktriangleleft$

With Lemma 8 at hand, we show that the expected size of the  $t$ -ball around the origin grows at most exponentially with  $t$ . We define this ball  $\mathcal{B}(x, t)$  as the set of vertices reachable from vertex  $x$  with a path of cost-distance at most  $t$ . Exponential growth is not enough by itself for our goal of showing a polylogarithmic lower bound on the distances but is a crucial step in doing so. To do this, we modify the proof of Lemma 2.6 and Theorem 1.2 (ii) in [13]. In the following, we only consider the growth of  $\mathcal{B}(0, t)$ , i.e., the  $t$ -ball around the origin, but it is easy to see that (by translation invariance) the same statements hold if we replace the origin by any vertex  $x$ .

**► Theorem 9 (Exponential Ball Growth).** *Let  $\mathcal{B}(0, t)$  denote the set of vertices reachable with a path of cost at most  $t$  from the origin in CFFP. If  $2\alpha < \tau - 1$ , we have for some  $C$  depending only on  $\alpha$  and  $\tau$ ,*

$$\mathbb{E}[|\mathcal{B}(0, t)|] \leq e^{Ct}.$$

**Proof.** We compute

$$\begin{aligned} \mathbb{E} [|\mathcal{B}(0, t)|] &= \sum_{u \in \mathbb{Z}^d} \mathbb{P} (d_G^{\text{cost}}(0, u) \leq t) \\ &\leq 1 + \sum_{\substack{u \in \mathbb{Z}^d \\ u \neq 0}} \sum_{k=1}^{\infty} \sum_{\substack{(0,u)\text{-path } \pi \\ \text{of length } k}} \mathbb{P} (\pi \text{ has cost distance at most } t) \\ &\stackrel{\text{Lemma 8}}{\leq} 1 + \sum_{\substack{u \in \mathbb{Z}^d \\ u \neq 0}} \sum_{k=1}^{\infty} \left(\frac{ect}{k}\right)^k \sum_{\substack{(0,u)\text{-path} \\ (0=u_1, u_2, \dots, u_{k+1}=u)}} \left[ \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d} \right]. \end{aligned}$$

The constant  $c$  above is as in Lemma 8. The rightmost sum above can be bounded by  $b^k |u|^{-\alpha d}$  for some  $b$  depending only on  $\alpha$ . This is done by Lemma 2.5 (c) in [13] (the quantity bounded there is the above sum and is defined in equation (2.3) in the page previous to that of Lemma 2.5). With that in mind, we have

$$\mathbb{E} [|\mathcal{B}(0, t)|] \leq 1 + \left( \sum_{u \in \mathbb{Z}^d, u \neq 0} |u|^{-\alpha d} \right) \left( \sum_{k=1}^{\infty} \left(\frac{ecbt}{k}\right)^k \right).$$

Since  $\alpha > 1$ , the first sum above is bounded by a constant  $c_1$ . Moreover, note that

$$\sum_{k=1}^{\infty} \left(\frac{ecbt}{k}\right)^k \leq \sum_{k=0}^{\infty} \frac{(ecbt)^k}{k!} - 1 = e^{ecbt} - 1.$$

One can choose  $C$  large enough so that  $\mathbb{E} [|\mathcal{B}(0, t)|] \leq e^{Ct}$ . To see why, note that we can freely assume  $c_1 \geq 1$ . Then, setting  $C = ecbc_1$  suffices. That is because of the following. Let  $f(x) = x^{c_1} + c_1(1-x) - 1$ . This function is decreasing from 0 to 1 and increasing afterwards. Moreover, both  $f(0)$  and  $f(1)$  are non-negative, hence it is non-negative for all  $x \geq 0$ . Setting  $x = e^{ecbt}$  shows that for all  $t \geq 0$ ,

$$\mathbb{E} [|\mathcal{B}(0, t)|] \leq 1 + c_1(e^{ecbt} - 1) \leq (e^{ecbt})^{c_1} = e^{Ct}. \quad \blacktriangleleft$$

In the following, we define

$$g(t) := \mathbb{E} [|\mathcal{B}(0, t)|]$$

and note that we have already shown that  $g(t)$  grows at most exponentially. But we can do better and show that in fact it grows at most *stretched* exponentially, in particular roughly as  $\exp(t^{\frac{1}{\Delta}})$ . This intuitively corresponds to the cost-distances between two vertices  $u$  and  $v$  growing roughly as  $(\log |u - v|)^{\Delta}$ , and is then also used in proving the corresponding lower bound later.

To show this improved bound, we bound the crucial quantity

$$f(r, t) = \sup_{|u|=r} \mathbb{P} (d_G^{\text{cost}}(0, u) \leq t) \in [0, 1],$$

that is, the highest possible probability with which a vertex connects to the origin with cost at most  $t$ , given that it has geometric distance  $r$ . We only consider  $r, t > 0$ . One can show the following bound for  $f(r, t)$ .

► **Lemma 10** (Towards a Self-Bounding Inequality for  $g(t)$ ). *Consider CFFP with  $2\alpha < \tau - 1$ . There exist constants  $c_f, \delta > 0$  depending only on  $\alpha$  and  $\tau$  such that*

$$f(r, t) \leq c_f r^{-\alpha d} h(t), \text{ where } h(t) := t^{\alpha d} \int_0^t g(t-y)(g(y) - 1) dy + e^{-\delta t}.$$

This lemma can be seen as a generalization of the technique we use to prove Lemma 4: We bound the probability that two vertices at distance  $r$  are connected by a path of cost at most  $t$  by a term that is essentially  $r^{-\alpha d}$  (which is roughly the probability that the longest edge in such a path exists) times  $h(t)$  which integrates over all possible  $y$  such that said edge connects the  $(t - y)$ -ball around 0 and the  $y$ -ball around  $u$ . Using this, we can then derive a *self-bounding inequality* for  $g(t)$ , which relates  $g$  recursively to itself such that we can derive an upper bound on  $g$  by solving said recursive relation using Theorem 13 which is identical to [13, Theorem 5.3]. We derive the self-bounding inequality by summing  $f(r, t)$  over all vertices and thus express  $g(t)$  as a function of  $h(t)$ , which – in turn – depends on  $g$ . We capture this in the following lemma.

► **Lemma 11 (Self-Bounding Inequality for  $g(t)$ ).** *Consider CFFP with  $2\alpha \leq \tau - 1$ . There are constants  $c, \delta$  such that for all  $t \geq 0$*

$$g(t)^\alpha \leq c \left( t^{\alpha d} \int_0^t g(t - y)g(y)dy + 1 \right).$$

**Proof.** To derive the self-bounding inequality for  $g$  using Lemma 10, we upper bound  $g$  by an expression involving  $f(r, t)$  and then upper bound  $f(r, t)$  using Lemma 10. Specifically, we estimate the expected size of a  $t$ -ball by integrating over all vertices times the respective probability  $f(r, t)$ .

$$\begin{aligned} \mathbb{E}[\mathcal{B}(0, t)] = g(t) &\leq 1 + \int_1^\infty c_d r^{d-1} \min\{1, f(r, t)\} dr \\ &= 1 + \int_1^{(c_f h(t))^{\frac{1}{\alpha d}}} c_d r^{d-1} dr + \int_{(c_f h(t))^{\frac{1}{\alpha d}}}^\infty c_d c_f r^{d-1-\alpha d} h(t) dr \end{aligned}$$

since for  $r > (c_f h(t))^{\frac{1}{\alpha d}}$ , the minimum is smaller than 1 by definition of  $f(r, t)$  from Lemma 10. Integrating out then yields,

$$g(t) \leq 1 + c' \left( h(t)^{\frac{1}{\alpha}} + h(t)^{\frac{1}{\alpha}-1} h(t) \right) \leq 1 + ((c'' h(t))^{\frac{1}{\alpha}})$$

for some constants  $c', c''$  that depend on  $\alpha, d$  and  $\tau$ . Therefore, we infer that

$$(g(t) - 1)^\alpha \leq c'' h(t). \tag{7}$$

It can be shown that<sup>12</sup>  $g(t)^\alpha \leq 2^{\alpha-1}(1 + (g(t) - 1)^\alpha)$ . Chaining this inequality with (7) and replacing  $h(t)$  above by its definition in Lemma 10 we get the claimed recursive inequality for  $g(t)$ . In more detail, we have

$$g(t)^\alpha \leq 2^{\alpha-1}(1 + (g(t) - 1)^\alpha) \leq 2^{\alpha-1}(1 + c'' h(t))$$

Since  $h(t)$  is bounded away from zero and since  $e^{-\delta t} \leq 1$ , it follows that there exists a  $c$  such that the inequality claimed in the lemma statement holds for all  $t$ . ◀

It is through this inequality that a stronger bound on  $g(t)$  can be derived. For this, we use Theorem 5.3 from [13] directly which we restate as Theorem 13. It claims (among more general things) that for a given function  $g(t)$ , if  $1 \leq g(t) \leq e^{Ct}$  for some constant  $C$  (which we have already shown) and an inequality similar to 7 holds, then one *roughly* has  $g(t) \leq e^{t^{1/\Delta}}$ . Now that we have motivated Lemma 10, let us prove it.

---

<sup>12</sup>One simply needs to consider the function  $f(x) = 2^{\alpha-1}(1 + (x - 1)^\alpha) - x^\alpha$  restricted to  $x \geq 1$ , which has a global minimum of 0 at  $x = 2$ .

**Proof of Lemma 10.** Fix a “target” vertex  $u$ . Let us first focus on paths from 0 to  $u$  which contain at least  $\rho t$  edges (we assume that this is an integer for simplicity) for some constant  $\rho$  that will be determined later. If  $P_{\text{long}}$  is the probability that some such path has cost-distance less than  $t$ , then by a simple union bound we have

$$\begin{aligned} P_{\text{long}} &\leq \sum_{k=\rho t}^{\infty} \sum_{\substack{(0,u)\text{-path} \\ \text{of length } k}} \mathbb{P}(\pi \text{ has cost-distance at most } t) \\ &\stackrel{\text{Lemma 8}}{\leq} e^{\theta t} \sum_{k=\rho t}^{\infty} \theta^{-k} c^k \sum_{\substack{(0,u)\text{-path} \\ \text{with } u_1=0 \text{ and } u_{k+1}=u}} \prod_{i=1}^k |u_i - u_{i+1}|^{-\alpha d} \\ &\stackrel{\text{Lemma 2.5 (c) in [13]}}{\leq} e^{\theta t} |u|^{-\alpha d} \sum_{k=\rho t}^{\infty} \left(\frac{cb}{\theta}\right)^k. \end{aligned}$$

In the second line above, we used Lemma 8 but the final step where  $\theta$  is set to some value is not carried out. Furthermore, the constant  $b$  that emerges in the third line is as in Lemma 2.5 (c) of [13]. Now, setting  $\theta = \rho > ecb$ , we have

$$P_{\text{long}} \leq |u|^{-\alpha d} e^{\rho t} \frac{\left(\frac{cb}{\rho}\right)^{\rho t}}{1 - \frac{cb}{\rho}} \leq |u|^{-\alpha d} \frac{e}{e-1} e^{-\rho t \log \frac{\rho}{ecb}}. \quad (8)$$

Now, let us turn our attention to paths that use at most  $\rho t$  edges instead and let  $P_{\text{short}}$  denote the probability that such a path has cost-distance at most  $t$ . The idea here is to notice that a geometrically long edge  $(u_1, u_2)$  must be used (similarly as in the proof of Lemma 4). In particular, this edge has to cover a distance of at least  $\frac{|u|}{\rho t}$ , as there are at most  $\rho t$  edges used to cover a distance of  $|u|$ . We adapt the argumentation of the proof of Lemma 5.1 in [13], which is essentially a union bound over all the possible intermediate pairs  $(u_1, u_2)$ . More precisely, we get that

$$\begin{aligned} P_{\text{short}} &\leq \sum_{\substack{u_1, u_2 \in \mathbb{Z}^d \\ |u_1 - u_2| \geq \frac{|u|}{\rho t}}} \iint \mathbb{P}(d_G^{\text{cost}}(0, u_1) + c_{(u_1, u_2)} + d_G^{\text{cost}}(u_2, u) \leq t \mid w_{u_1}, w_{u_2}) d\mu(w_{u_1}) d\mu(w_{u_2}) \end{aligned} \quad (9)$$

where  $\mu(w) = w^{1-\tau}$  is the probability measure of the weight distribution and

$$\begin{aligned} &\mathbb{P}(d_G^{\text{cost}}(0, u_1) + c_{(u_1, u_2)} + d_G^{\text{cost}}(u_2, u) \leq t \mid w_{u_1}, w_{u_2}) \\ &\leq \int_0^t d\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq s \mid w_{u_1}) \int_0^{t-s} \mathbb{P}(d_G^{\text{cost}}(u_2, u) \leq y \mid w_{u_2}) w_{u_1}^\alpha w_{u_2}^\alpha |u|^{-d\alpha} (\rho t)^{\alpha d} dy. \end{aligned} \quad (10)$$

where we use a convolution over the cost of the left and right path segment and that the density of  $c_{(u_1, u_2)}$  is at most  $w_{u_1}^\alpha w_{u_2}^\alpha |u|^{-d\alpha} (\rho t)^{\alpha d}$ . Had there not been weights involved, the proof of Lemma 5.1 in [13] would show immediately that

$$P_{\text{short}} \leq c_{\text{short}} |u|^{-\alpha d} h(t)$$

for some constant  $c_{\text{short}}$  depending on  $\alpha$ . In our case we first need to get rid of the weights. To this end, we use the following simple proposition whose proof we defer for now.

► **Proposition 12.** For any  $u_1 \in \mathbb{Z}^d$  and any  $t > 0$ , we have

$$\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq t \mid w_1) \leq 2w_1^\alpha \mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq t).$$

Applying this proposition to (10) yields

$$\begin{aligned} & \mathbb{P}(d_G^{\text{cost}}(0, u_1) + c_{(u_1, u_2)} + d_G^{\text{cost}}(u_2, u) \leq t \mid w_{u_1}, w_{u_2}) \\ & \leq 4w_{u_1}^{2\alpha} w_{u_2}^{2\alpha} |u|^{-d\alpha} (\rho t)^{\alpha d} \int_0^t d\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq s) \int_0^{t-s} \mathbb{P}(d_G^{\text{cost}}(u_2, u) \leq y) dy. \end{aligned}$$

Then, applying this to (9) and taking the sum into the integrals, we get

$$\begin{aligned} P_{\text{short}} & \leq \sum_{\substack{u_1, u_2 \in \mathbb{Z}^d \\ |u_1 - u_2| \geq |u|/(\rho t)}} \left( \iint 4w_{u_1}^{2\alpha} w_{u_2}^{2\alpha} d\mu(w_{u_1}) d\mu(w_{u_2}) \right) \\ & \quad \times \left( |u|^{-d\alpha} (\rho t)^{\alpha d} \int_0^t d\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq s) \int_0^{t-s} \mathbb{P}(d_G^{\text{cost}}(u_2, u) \leq y) dy \right) \\ & \leq \left( \int_1^\infty \int_1^\infty 4(\tau - 1)^2 w_{u_1}^{2\alpha - \tau} w_{u_2}^{2\alpha - \tau} dw_{u_1} dw_{u_2} \right) \\ & \quad \times \left( |u|^{-d\alpha} (\rho t)^{\alpha d} \sum_{u_1 \in \mathbb{Z}^d} \sum_{u_2 \in \mathbb{Z}^d} \int_0^t d\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq s) \int_0^{t-s} \mathbb{P}(d_G^{\text{cost}}(u_2, u) \leq y) dy \right) \end{aligned}$$

where we took the sum into the second set of parentheses and omitted the condition  $|u_1 - u_2| \geq |u|/(\rho t)$  from the sum and then split it into two sums. We also replaced  $\mu(w) = 1 - w^{1-\tau}$  so  $d\mu(w) = (\tau - 1)w^{-\tau} dw$ . Now, taking these sums into the integrals and recalling that  $g(x) = \sum_{v \in \mathbb{Z}^d} \mathbb{P}(d_G^{\text{cost}}(0, v) \leq x)$  yields that

$$P_{\text{short}} \leq \left( \int_1^\infty \int_1^\infty 4w_{u_1}^{2\alpha - \tau} w_{u_2}^{2\alpha - \tau} dw_{u_1} dw_{u_2} \right) \left( |u|^{-d\alpha} (\rho t)^{\alpha d} \int_0^t dg(s) \int_0^{t-s} g(y) dy \right).$$

Here, it is easy to see that the term in the second set of parentheses is at most  $c_{\text{short}} |u|^{-\alpha d} h(t)$  for some constant  $c_{\text{short}}$  as was formally shown in [13, Lemma 5.1]. The term in the first set of parentheses is some constant that only depends on  $\alpha, \tau$  since the condition  $2\alpha < \tau - 1$  ensures that the integrals converge. This constant enters into  $c_{\text{short}}$ .

If we now choose  $\rho > ecb$  and  $\delta = \rho \log \frac{\rho}{ecb}$ , then summing  $P_{\text{long}}$  from (8) and  $P_{\text{short}}$  as above yields

$$f(r, t) \leq c_f r^{-\alpha d} h(t)$$

for some constant  $c_f$  as desired. ◀

We now give the proof of the proposition deferred above.

**Proof of Proposition 12.** Recall that the statement of the proposition is that

$$\mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq t \mid w_1) \leq 2w_1^\alpha \mathbb{P}(d_G^{\text{cost}}(0, u_1) \leq t)$$

We show this using a coupling argument. To this end, we consider a model  $M$  that resembles CFFP where instead of having the vertex  $u_1$  with weight  $w_1$ , we set the weight of  $u_1$  *deterministically* equal to 1. However, to decide the costs of edges adjacent to  $u_1$ , we take the minimum of  $\lceil w_1^\alpha \rceil$  many independent samples, i.e., for the edge  $(u_1, v)$  we set its cost to the minimum of  $\lceil w_1^\alpha \rceil$  independent samples from an exponential distribution with



rate  $w_v^\alpha |u - v|^{-\alpha d}$ . Alternatively, this can be seen as having  $\lceil w_1^\alpha \rceil$  copies of  $u_1$  (with weight 1) and then studying the minimal cost-distance from 0 to one of the copies  $u_1$  where the minimum is taken over all copies. Note that this description diverges from the one in the previous sentence if we are interested in paths between two arbitrary vertices, but since we are only concerned with paths from 0 to  $u_1$ , the two are equivalent.

We continue by showing that cost distances in CFFP (from 0 to  $u_1$ ) stochastically dominate those in  $M$ , i.e., that

$$\mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in CFFP} \mid w_1) \leq \mathbb{P}(d_G^{\text{cost}}(u'_1, 0) \leq t \text{ in } M).$$

To this end, we first show that this is the case for the cost of all edges incident to  $u_1$ . Let  $e = (u_1, v)$  be an arbitrary such edge and note that

$$\begin{aligned} \mathbb{P}(c_{(u_1, v)} \geq x \text{ in } M) &= \left( e^{-x w_v^\alpha |u_1 - v|^{-\alpha d}} \right)^{\lceil w_1^\alpha \rceil} \\ &\leq e^{-x w_1^\alpha w_v^\alpha |u_1 - v|^{-\alpha d}} = \mathbb{P}(c_{(u_1, v)} \geq x \text{ in CFFP}). \end{aligned}$$

Now, let  $\mathcal{R}$  be a fixed realization of all weights and edge costs in CFFP not associated with  $u_1$  and denote by  $[d_G^{\text{cost}}(0, v)]_{\mathcal{R}}$  the cost distance from 0 to  $v$  in this realization when  $u_1$  is removed from the underlying graph. With this, we note that – conditional on  $\mathcal{R}$  – we have

$$d_G^{\text{cost}}(u_1, 0) \leq t \iff \exists v \text{ such that } c_{(u_1, v)} \leq t - [d_G^{\text{cost}}(0, v)]_{\mathcal{R}}.$$

Since  $\mathbb{P}(c_{(v, u_1)} \geq x \text{ in } M) \leq \mathbb{P}(c_{(v, u_1)} \geq x \text{ in CFFP})$  as shown above, the probability that this occurs in  $M$  is at least as large as the corresponding probability in CFFP, and since this holds conditional on any realization  $\mathcal{R}$ , it also holds unconditionally by the law of total probability. Hence, we have shown the desired stochastic domination.

It remains to be shown that

$$\mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t' \text{ in } M) \leq 2w_1^\alpha \mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in CFFP}).$$

To prove this, we again consider an arbitrary but fixed realization  $\mathcal{R}$  as in the previous paragraph and recall that  $d_G^{\text{cost}}(u_1, 0) \leq t$  if and only if there is some  $v$  such that  $c_{(u_1, v)} \leq t - [d_G^{\text{cost}}(0, v)]_{\mathcal{R}}$ . Note that by the definition of  $M$ , this occurs if and only if it happens for at least one of the  $\lceil w_1^\alpha \rceil \leq 2w_1^\alpha$  copies of  $u_1$ . Since for each copy, the probability that this happens for said copy is  $\mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in CFFP} \mid w_{u_1} = 1, \mathcal{R})$ , we get from a union bound and from the law of total probability that

$$\begin{aligned} \mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in } M) &\leq 2w_1^\alpha \mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in CFFP} \mid w_{u_1} = 1) \\ &\leq 2w_1^\alpha \mathbb{P}(d_G^{\text{cost}}(u_1, 0) \leq t \text{ in CFFP}) \quad \blacktriangleleft \end{aligned}$$

Finally, we will use Lemma 10 in conjunction with Theorem 5.3 from [13] to show the desired lower bound on cost-distances. We restate that theorem here, simplified for our use case.

► **Theorem 13** (Theorem 5.3 from [13]). *Let  $g(t) : [0, \infty) \rightarrow \mathbb{R}$  be a function satisfying*

$$1 \leq g(t) \leq e^{Ct}$$

and

$$g(t)^{1/\theta} \leq c_h \left( 1 + t^{\beta-1} \int_0^t g(y)g(t-y) dy \right)$$

for all  $t \geq 0$  for some constants  $C > 0$ ,  $\theta \in (\frac{1}{2}, 1)$ ,  $\beta \geq 0$ , and  $c_h \geq 1$ . Then, there exists a constant  $c_\theta > 1$  such that  $g(t) \leq G(t)$  for all  $t \geq 0$  where  $G(t)$  is defined such that

$$\log G(t) = c_\theta (2\lambda t)^{\log_2(2^\theta)} (\log(1 + t^\beta))^{\log_2(1/\theta)} (1 + o(1)).$$

We use the above theorem to prove our main lemma for FPP, which we restate here.

► **Lemma 14** (Tail Bound for Cost-Distances in FPP). *Consider FPP on SFP with  $2\alpha < \tau - 1$  and arbitrary vertices  $x, y$ . There exists a constant  $c$  depending only on  $\alpha$  and  $\tau$  such that for  $\Delta = \Delta(\alpha) = 1/\log_2(2/\alpha)$ ,*

$$\log \mathbb{P}(d_G^{\text{cost}}(x, y) \leq t) \leq c(\log(1 + t))^{1-1/\Delta} t^{1/\Delta} (1 + o(1)) - \alpha d \log |x - y| + c.$$

**Proof.** As discussed, it suffices to show the claim for CFFP. Moreover, by translation invariance, we can replace  $x$  with the origin 0 and  $y$  by  $u = y - x$ . By Lemma 11 we have

$$g(t)^\alpha \leq c_1 \left( t^{\alpha d} \int_0^t g(t-y)g(y) dy + 1 \right) \tag{11}$$

for some  $c_1, \delta > 0$  depending only on parameters of the model. We also know that  $g(t) \leq e^{Ct}$  from Theorem 9 for a  $C$  with the same dependencies. Therefore, using Theorem 5.3 from [13] (stated above the current theorem) with  $\theta = \frac{1}{\alpha}, \beta = \alpha d + 1$  and with the inequality  $1 + t^\beta \leq (1 + t)^\beta$ , we have  $g(t) \leq G(t)$ , where

$$\log G(t) = c(\log(1 + t))^{1-\frac{1}{\Delta}} t^{\frac{1}{\Delta}} (1 + o(1)),$$

where  $c$  depends only on parameters of the model. Now, by Lemma 10, we have

$$\begin{aligned} \log \mathbb{P}(d_G^{\text{cost}}(0, u) \leq t) &\leq \log f(|u|, t) \\ &\leq c' - \alpha d \log |u| + \log \left( t^{\alpha d} \int_0^t g(t-y)(g(y) - 1) dy + e^{-\delta t} \right) \end{aligned}$$

The constant  $c'$  above comes from Lemma 10. The final step is to notice that inequality (11) is satisfied as equality for  $G(t)$ , as in [13]. Then, substituting the expression for it in the resulting inequality gives the desired bound. ◀

We can use the coupling employed in Theorem 1.1 to establish the above tail bound even if  $2\alpha \geq \tau - 1$  but with  $\Delta$  replaced by  $\Delta'' = \Delta(\min\{\alpha, \frac{\tau-1}{2}\} - \varepsilon)$  for arbitrarily small  $\varepsilon$ . We then have obtained Theorem 1.3. Since the proof is almost verbatim the same as that of Theorem 1.1, we omit it. Note that one minor technical step that is required additionally to get the bound claimed in Theorem 1.3 using the one obtained from Lemma 14 is to introduce an auxiliary constant  $\varepsilon'$  in addition to the  $\varepsilon$  from the statement of the theorem. This swallows the  $(\log(1 + t))^{1-\frac{1}{\Delta}}$  factor.

---

**References**

- 1 Mohammed Amin Abdullah, Michel Bode, and Nikolaos Fountoulakis. Typical distances in a geometric model for complex networks. *Internet Mathematics*, 1:38, 2017.
- 2 Noam Berger. A lower bound for the chemical distance in sparse long-range percolation models. *arXiv preprint*, 2004. [arXiv:math/0409021](https://arxiv.org/abs/math/0409021).
- 3 Marek Biskup. On the scaling of the chemical distance in long-range percolation models. *The Annals of Probability*, 32(4):2938–2977, October 2004. doi:10.1214/009117904000000577.

- 4 Marek Biskup. Graph diameter in long-range percolation. *Random Structures & Algorithms*, 39(2):210–227, 2011. doi:10.1002/rsa.20349.
- 5 Marek Biskup and Jeffrey Lin. Sharp asymptotic for the chemical distance in long-range percolation. *Random Structures & Algorithms*, 55(3):560–583, 2019. doi:10.1002/rsa.20849.
- 6 Thomas Bläsius and Philipp Fischbeck. On the external validity of average-case analyses of graph algorithms. *ACM Transactions on Algorithms*, 20(1):1–42, 2024.
- 7 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, and Anton Krohmer. Hyperbolic embeddings for near-optimal greedy routing. *ACM J. Exp. Algorithmics*, 25, 2020. doi:10.1145/3381751.
- 8 Thomas Bläsius, Tobias Friedrich, and Christopher Weyand. Efficiently computing maximum flows in scale-free networks. *arXiv preprint*, 2020. arXiv:2009.09678.
- 9 Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):62, 2010.
- 10 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Average distance in a general class of scale-free networks with underlying geometry. *arXiv preprint*, 2016. arXiv:1602.05712.
- 11 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theoretical Computer Science*, 760:35–54, 2019.
- 12 Karl Bringmann, Ralph Keusch, Johannes Lengler, Yannic Maus, and Anisur R Molla. Greedy routing and the algorithmic small-world phenomenon. *Journal of Computer and System Sciences*, 125:59–105, 2022.
- 13 Shirshendu Chatterjee and Partha S. Dey. Multiple phase transitions in long-range first-passage percolation on square lattices. *Communications on Pure and Applied Mathematics*, 69(2):203–256, 2016. doi:10.1002/cpa.21571.
- 14 Maria Deijfen, Remco Hofstad, and Gerard Hooghiemstra. Scale-free percolation. *Annales de l’Institut Henri Poincaré Probabilités et Statistiques*, 49:817–838, August 2013. doi:10.1214/12-AIHP480.
- 15 Philippe Deprez, Rajat Subhra Hazra, and Mario V Wüthrich. Inhomogeneous long-range percolation for real-life network modeling. *Risks*, 3(1):1–23, 2015.
- 16 Philippe Deprez and Mario V Wüthrich. Scale-free percolation in continuum space. *Communications in Mathematics and Statistics*, 7(3):269–308, 2019.
- 17 Leslie Ann Goldberg, Joost Jorritsma, Júlia Komjáthy, and John Lapinskas. Increasing efficacy of contact-tracing applications by user referrals and stricter quarantining. *Plos one*, 16(5):e0250435, 2021.
- 18 Mark S Granovetter. The strength of weak ties. *American journal of sociology*, 78(6):1360–1380, 1973.
- 19 Nannan Hao and Markus Heydenreich. Graph distances in scale-free percolation: the logarithmic case. *Journal of Applied Probability*, 60(1):295–313, 2023.
- 20 Joost Jorritsma, Tim Hulshof, and Júlia Komjáthy. Not all interventions are equal for the height of the second peak. *Chaos, Solitons & Fractals*, 139:109965, 2020.
- 21 Christoph Koch and Johannes Lengler. Bootstrap percolation on geometric inhomogeneous random graphs. *Internet Mathematics*, page 18995, 2021.
- 22 Júlia Komjáthy, John Lapinskas, and Johannes Lengler. Penalising transmission to hubs in scale-free spatial random graphs. *Annales de l’Institut Henri Poincaré, Probabilités et Statistiques*, 57(4):1968–2016, 2021. doi:10.1214/21-AIHP1149.
- 23 Júlia Komjáthy, John Lapinskas, Johannes Lengler, and Ulysse Schaller. Four universal growth regimes in degree-dependent first passage percolation on spatial random graphs i. *arXiv preprint*, 2023. arXiv:2309.11840.
- 24 Júlia Komjáthy, John Lapinskas, Johannes Lengler, and Ulysse Schaller. Four universal growth regimes in degree-dependent first passage percolation on spatial random graphs ii. *arXiv preprint*, 2023. arXiv:230911880.

- 25 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.
- 26 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.
- 27 Gergely Ódor, Domonkos Czifra, Júlia Komjáthy, László Lovász, and Márton Karsai. Switchover phenomenon induced by epidemic seeding on geometric networks. *Proceedings of the National Academy of Sciences*, 118(41):e2112607118, 2021.
- 28 David Reimer. Proof of the van den berg–kesten conjecture. *Combinatorics, Probability and Computing*, 9(1):27–32, 2000. doi:10.1017/S0963548399004113.
- 29 Pieter Trapman. The growth of the infinite long-range percolation cluster. *The Annals of Probability*, 38(4):1583–1608, 2010.

### **A** On the upper bound claim in [19]

In this section, we briefly explain the mistake in the upper bound proof for graph distances in [19]. There, the authors add edges for some paths of length 2 in SFP, thus cutting graph distances at most in half. Let us call the resulting graph 2-SFP. Then they compare 2-SFP to LRP with different parameters. The 2-SFP graph does not dominate an LRP because the edges are correlated, but for every edge  $e$ ,  $\mathbb{P}(e \text{ open in 2-SFP}) \geq \mathbb{P}(e \text{ open in LRP})$ .

Then they make the argument that the correlations are positive, so that for every fixed  $x$ - $y$ -path  $\pi$ , by the FKG inequality:

$$\mathbb{P}(\pi \text{ open in 2-SFP}) \geq \mathbb{P}(\pi \text{ open in LRP}). \quad (12)$$

That is correct, but it does not imply the statement that we would want for a suitable  $k$ :

$$\begin{aligned} \mathbb{P}(\exists x\text{-}y\text{-path } \pi \text{ of length } \leq k: \pi \text{ open in 2-SFP}) \\ \geq \mathbb{P}(\exists x\text{-}y\text{-path } \pi \text{ of length } \leq k: \pi \text{ open in LRP}). \end{aligned} \quad (13)$$

Instead, (12) only implies by summing over all  $x$ - $y$ -paths of length at most  $k$ :

$$\mathbb{E}[\#\text{ of open } x\text{-}y\text{-paths of length } \leq k \text{ in 2-SFP}] \quad (14)$$

$$\geq \mathbb{E}[\#\text{ of open } x\text{-}y\text{-paths of length } \leq k \text{ in LRP}]. \quad (15)$$

However, it is not hard to see that due to the correlations the left hand side of (14) is dominated by low-probability events where the number of paths is very large. In particular, the upper bound proof for LRP is centered around the concept of *hierarchies*, and the first step of a hierarchy is to find an edge of length  $\Theta(|x - y|)$ , where the two endpoints lie respectively close to  $x$  and  $y$ . It is easy to see that for some parameters considered in [19], with high probability such edges do exist in LRP but do not exist in 2-SFP. However, the *expected number* of such edges in 2-SFP (corresponding to the left hand side of (14)) is still large because the unlikely event of a vertex of weight  $\Theta(|x - y|)$  at distance  $|x - y|$  from  $x$  induces a very large number of 2-paths in SFP, which become edges in 2-SFP.

Hence, the argument in [19] shows (14) but not (13), and this is not a minor omission but a major gap. In fact, we conjecture that the upper bound statements in [19] are false, and that the exponent  $\Delta(\alpha) = 1/\log_2(2/\alpha)$  is tight throughout the polylogarithmic regime, i.e. for all  $\tau > 3$  and  $\alpha \in (1, 2)$ . The reason for this intuition is that for  $\tau > 3$  there exists a constant  $C > 0$  such that the induced graph of vertices of weight larger than  $C$  does not percolate. However, we do not see an obvious way to leverage this property into lower bounds for graph distances.

# Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

Pranjal Dutta   

School of Computing, National University of Singapore, Singapore

Amit Sinhababu 

Chennai Mathematical Institute, Chennai, India

Thomas Thierauf  

Ulm University, Germany

---

## Abstract

Kaltofen [STOC 1986] gave a randomized algorithm to factor multivariate polynomials given by algebraic circuits. We derandomize the algorithm in some special cases.

For an  $n$ -variate polynomial  $f$  of degree  $d$  from a class  $\mathcal{C}$  of algebraic circuits, we design a deterministic algorithm to find all its irreducible factors of degree  $\leq \delta$ , for *constant*  $\delta$ . The running time of this algorithm stems from a deterministic PIT algorithm for class  $\mathcal{C}$  and a deterministic algorithm that tests divisibility of  $f$  by a polynomial of degree  $\leq \delta$ .

By using the PIT algorithm for constant-depth circuits by Limaye, Srinivasan and Tavenas [FOCS 2021] and the divisibility results by Forbes [FOCS 2015], this generalizes and simplifies a recent result by Kumar, Ramanathan and Saptharishi [SODA 2024]. They designed a subexponential-time algorithm that, given a blackbox access to  $f$  computed by a constant-depth circuit, outputs its irreducible factors of degree  $\leq \delta$ . When the input  $f$  is sparse, the time complexity of our algorithm depends on a whitebox PIT algorithm for  $\sum_i m_i g_i^{d_i}$ , where  $m_i$  are monomials and  $\deg(g_i) \leq \delta$ . All the previous algorithms required a blackbox PIT algorithm for the same class.

Our second main result considers polynomials  $f$ , where *each* irreducible factor has degree at most  $\delta$ . We show that all the irreducible factors with their multiplicities can be computed in polynomial time with blackbox access to  $f$ .

Finally, we consider factorization of *sparse* polynomials. We show that in order to compute all the sparse irreducible factors efficiently, it suffices to derandomize irreducibility preserving bivariate projections for sparse polynomials.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Algebraic complexity theory; Computing methodologies  $\rightarrow$  Algebraic algorithms

**Keywords and phrases** algebraic complexity, factoring, low degree, weight isolation, divisibility

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2024.75

**Category** RANDOM

**Funding** *Pranjal Dutta*: Funded by the project “Foundation of Lattice-based Cryptography”, by NUS-NCS Joint Laboratory for Cyber Security.

*Thomas Thierauf*: Supported by DFG grant TH 472/5-2.

## 1 Introduction

The problem of *multivariate polynomial factorization* asks to find the unique factorization of a given polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  as a product of distinct irreducible polynomials over  $\mathbb{F}$ . The problem reduces to *univariate polynomial factorization* over the same field, for which a deterministic polynomial time algorithm is known over the field  $\mathbb{Q}$ . The complexity of multivariate factorization depends on the representation of input and output polynomials. If we use *dense representation* (where all the coefficients are listed including the zero coefficients), deterministic polynomial time algorithms for multivariate factoring



© Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024).

Editors: Amit Kumar and Noga Ron-Zewi; Article No. 75; pp. 75:1–75:20



Leibniz International Proceedings in Informatics

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

are known [16]. If we use *sparse representation* (where only the nonzero coefficients are listed), only randomized polynomial time (in the total sparsity of input polynomial and the output factors) algorithms are known [41, 21]. There are other standard representations like arithmetic circuits, and blackbox models (that gives the evaluations of the polynomial at any point, but the internal structure of the computation is hidden). Randomized polynomial time factorization algorithms are known in these models due to the classic results of Kaltofen [19] and Kaltofen and Trager [21]. Randomization is naturally required for these models, as the more basic question of *polynomial identity testing* (given a circuit/blackbox, test if it computes the zero polynomial) is not yet derandomized.

Towards derandomization of special cases of multivariate factoring, we are motivated by the following two questions.

► **Question 1.** Given a sparse polynomial  $f$ . Can we find all the sparse irreducible factors of  $f$  by a deterministic algorithm in polynomial/quasipolynomial/subexponential time?

Note that the factors of a sparse polynomial  $f$  might be nonsparse. Bhargava, Saraf and Volkovich [3] showed an upper bound on the sparsity of the factors of a sparse polynomial  $f$ . However, the bound is exponential in the degree of  $f$ . Therefore, instead of finding *all* the irreducible factors, we want to output only those factors that are sparse.

► **Question 2 ([41, 3]).** Given polynomial  $f = \prod_{i=1}^m g_i^{e_i}$  as a blackbox, where polynomials  $g_i$  are irreducible polynomials whose sparsities are bounded by  $s$ . Can we find the polynomials  $g_i$  in *deterministic* time  $\text{poly}(s, n, d)$  or time quasi-poly/sub-exponential in  $s, n, d$ ?

The second question can be seen as a special case of polynomial factorization, where we are *promised* that *all* the irreducible factors are sparse. To our surprise, we do not know a deterministic *subexponential*-time algorithm even for the special case of Question 2, when the given blackbox computes the product of just *two* irreducible sparse polynomials.

Derandomization of multivariate factoring (whitebox, or blackbox) reduces to (whitebox, or correspondingly blackbox) derandomization of polynomial identity testing (PIT). Kopparty, Saraf and Shpilka [27] showed this reduction in the model of arithmetic circuits. However, we do not know if sparse factorization reduces to sparse PIT or constant-depth arithmetic circuit PIT (the algorithms of [27] reduce to general arithmetic circuit PIT). Recently, there has been some progress on these questions by [28, 29]. Earlier works of Volkovich [39, 40] made progress on several special cases of sparse multivariate factoring.

Multivariate polynomial factoring has various applications, such as *low-degree testing* [1], constructions of pseudorandom generators for low-degree polynomials [6, 8], computational algebraic geometry [14] and many more. blackbox multivariate polynomial factorization is extensively used in *arithmetic circuit reconstruction* [36, 37], and polynomial equivalence testing [22, 23, 33]. Algebraic hardness vs randomness [15] results crucially use multivariate factorization. Special cases of depth-4 polynomial identity testing are related to questions about sparse polynomial factorization [12, 40, 4].

**Divisibility testing.** In a factorization algorithm, we may want to check if a candidate factor is truly a factor via *divisibility testing*. It asks to test if a polynomial  $g(\mathbf{z})$  divides a polynomial  $f(\mathbf{z})$ . Forbes [9] showed that the divisibility testing question can be efficiently reduced to an instance of a PIT question of a model that relates to both  $f$  and  $g$ ; see Lemma 9. Currently, we do not know any deterministic polynomial time algorithm even when  $g$  and  $f$  are both sparse polynomials. When  $f$  is a sparse polynomial and  $g$  is a linear polynomial, the problem reduces to polynomial identity testing of any-order read-once oblivious branching programs (ROABPs), for which polynomial time whitebox PIT algorithm [34] and quasipolynomial

time blackbox PIT algorithms are known [10, 13, 11]. We do not know a deterministic polynomial time algorithm, even for testing if a quadratic polynomial  $g$  divides a sparse polynomial.

## 1.1 Our results

We show a general result that exhibits properties of a class  $\mathcal{C}$  of polynomials, such that we can compute the *constant-degree factors* of polynomials  $f \in \mathcal{C}$ . The following theorem is an informal statement of Theorem 18.

► **Theorem 1** (Low-degree factors via divisibility). *Let  $\delta \in \mathbb{N}$  be a constant and  $\mathcal{C}$  be a class of polynomials such that there is an efficient PIT algorithm for  $\mathcal{C}$ . For any  $n$ -variate polynomial  $f \in \mathcal{C}$  of degree  $d$ , finding all its irreducible factors of degree  $\leq \delta$  reduces to solving polynomially many divisibility questions of whether a given polynomial of degree  $\leq \delta$  divides  $f$ .*

Arguably, Theorem 1 generalizes and simplifies a recent result by Kumar, Ramanathan and Saptharishi [28] about the factorization of polynomials computed by constant-depth circuits. Importantly, if  $f$  is represented in the whitebox setting, then both the required algorithms (PIT and divisibility testing) in Theorem 1 are *whitebox* algorithms, whereas [28] still requires blackbox algorithms. We compare the results in more detail in Section 1.2.

We can apply Theorem 1 in the case of (any-order) ROABPs. There are polynomial time (respectively, quasipolynomial time) whitebox (respectively, blackbox) PIT algorithms for ROABPs. Moreover, using the divisibility techniques by Forbes [9] (see Lemma 9) and the duality trick by Saxena [35], the divisibility testing question of whether a given linear polynomial divides a ROABP can be reduced to a PIT instance of a polynomial-size ROABP.

► **Corollary 2** (Linear factors of ROABPs). *Let  $f$  be an  $n$ -variate polynomial of degree  $d$ , computed by an any-order ROABP of width  $w$ . Then one can output all its linear factors, along with the exponents in time  $\text{poly}(ndw)$  in the whitebox setting, and in time  $\text{poly}(ndw^{\log \log w})$  in the blackbox setting.*

When the input  $f$  is  $s$ -sparse, this result is already known due to Volkovich [39, Theorem 4]. Note that a sparse polynomial has a trivial ROABP.

Finally, we remark that Theorem 1 can be further generalized to outputting factors from a general class  $\mathcal{D}$  (as black box) when in addition to assuming (informally speaking) efficient PIT for  $\mathcal{C}$  and the divisibility test for  $\mathcal{C}$  by  $\mathcal{D}$ , one has to assume efficient derandomization of HIT for  $\mathcal{D}$  (in the sense of finding a good bivariate projection preserving irreducibility, see Assumption 1) and an inclusion property (i.e. given  $g \in \mathcal{D}$  or not). For simplicity, in the conference version, we only assume that  $\mathcal{D}$  is the class of constant-degree polynomials.

Our second result considers the class of polynomials  $f$ , where all the irreducible factors of  $f$  are promised to have degrees bounded by  $\delta$ . For this class, there are blackbox PIT algorithms with time complexity  $\text{poly}(d, n^\delta)$  for  $n$ -variate polynomials  $f$  of degree  $d$ , see [7, 5]. Hence, by Theorem 1, the factoring problem reduces to a divisibility question. Using techniques of Forbes [9] (see Lemma 9), this can be further reduced to designing a blackbox PIT algorithm for polynomials of the form  $\sum_i \Pi_j f_{i,j}$ , where  $\deg(f_{i,j}) \leq \delta$ . However, we *do not* know better than subexponential-time PIT algorithms for this model. Thus, Theorem 1 does not yield anything fruitful in this promise setting. We show how to completely avoid divisibility testing and still find all the irreducible factors in polynomial time. The following theorem is an informal statement of Theorem 19.

► **Theorem 3** (Promise low-degree factoring). *Let  $\delta \in \mathbb{N}$  be a constant. Given a blackbox access to an  $n$ -variate polynomial  $f$  of degree  $d$  such that all its irreducible factors have degrees at most  $\delta$ , one can deterministically output all its irreducible factors along with the multiplicities in  $\text{poly}(nd)$  time.*

The above theorem can be generalized to polynomials whose irreducible factors are from a class  $\mathcal{D}$  (and we will output them as blackbox), for which efficient PIT and derandomization of HIT (in the sense of finding a good bivariate projection preserving irreducibility; see Assumption 1) is known. For simplicity, in the conference version, we only focus on the class of constant-degree polynomials.

**Related work on Theorem 3.** There have been some works when  $\delta = 1$ . In this setting, given a promise that  $f(\mathbf{z}) = \prod_{i \in [m]} \ell_i^{e_i}$ , where  $\ell_i(\mathbf{z})$  are mutually co-prime linear polynomials, we have to output  $\ell_i$ . A randomized polynomial time algorithm for this problem follows from the work of Kaltofen and Trager [21]. Recently, Koiran and Ressayre [25] gave *three* different randomized algorithms for the *non-promise* problem that can test if a given  $f$  can be completely factored into linear polynomials and output the factorization if it exists. The first algorithm assumes that  $\ell_1, \dots, \ell_m$  are *linearly independent*, while the last two do not need that assumption. Later, Koiran and Skomra [26] derandomized the first algorithm when  $\ell_i$  are linearly independent. Using a different idea and linear independence of  $\ell_i$ , Medini and Shpilka [32] gave an alternative deterministic polynomial time algorithm. All these works exploited the linearity (and sometimes randomization/linear independence) of the factors, while our algorithm neither requires linearity of the factors nor any linear independence.

Finally, we go back to Question 1 of outputting all the sparse irreducible factors of a given sparse polynomial. Can efficient sparse irreducibility testing lead to an efficient algorithm? For general multivariate factoring, an effective version of Hilbert’s Irreducibility Theorem (HIT) by Kaltofen [17] (Theorem 10) says that with high probability, an irreducible  $n$ -variate polynomial remains irreducible if we randomly project it to a bivariate polynomial. This leads to an efficient factoring algorithm, since HIT helps to preserve the *factorization pattern* (the number of distinct irreducible factors and corresponding multiplicities). The hardness of Question 1 stems from the fact that a sparse polynomial may have both sparse and non-sparse irreducible factors. Hence, preserving irreducibility for sparse polynomials will not preserve the factorization pattern, and therefore, it may be hard to get back the actual factor. However, we observe that a deterministic version of HIT for sparse polynomials can indeed solve Question 1. The following theorem is an informal statement of Theorem 20.

► **Theorem 4** (Conditional sparse factoring, Informal). *Suppose there is an efficient algorithm that finds a bivariate projection, making an  $s$ -sparse irreducible polynomial both monic (in one variable) and irreducible. Then there is a subexponential-time algorithm that outputs all its irreducible factors with sparsities  $\leq s$  along with their multiplicities.*

## 1.2 Comparison with Kumar, Ramanathan and Saptharishi [28]

**Theorem 1 implies [28, Theorem 1.1–1.2].** Let  $\Delta \geq 2$  be an arbitrary positive integer. Assume that we have a blackbox access to  $f$ , which can be computed by a  $\Delta$ -depth algebraic circuit of size  $s$ . The recent breakthrough result of Limaye, Srinivasan, and Tavenas [31] gives a subexponential time identity testing algorithm for  $f$ . Moreover, using the techniques from [9] (see Lemma 9), one can show that whether a given polynomial of degree  $\leq \delta$  divides  $f$  can be efficiently reduced to PIT for an algebraic circuit of size  $\text{poly}(sd)$ , of the



form  $\sum_i g_i h_i^{d_i}$ , where the polynomials  $g_i$  are computable by  $\Delta$ -depth algebraic circuits and  $\deg(h_i) \leq \delta$ . For a formal proof, see [28, Corollary 2.19]. The main time complexity of [28, Theorem 1.1] is also dictated by the best-known blackbox PIT algorithm for the same model as above, which runs in subexponential time [31]. This algorithm requires the underlying field to have characteristic 0.

When  $\Delta = 2$ , Theorem 1 gives a quasipolynomial time algorithm to output irreducible factors of degree  $\leq \delta$ , thus implying [28, Theorem 1.2]. We know a polynomial time identity testing for sparse polynomials, due to Klivans and Spielman [24]. Further, [9, Corollary 7.16] showed that whether a polynomial of degree  $\leq \delta$  divides a sparse polynomial, reduces to PIT for  $\Sigma m \wedge \Sigma \Pi^{[\delta]}$ ; this model computes polynomials of the form  $\sum_{i=1}^{\text{poly}(sd)} m_i h_i^{d_i}$ , where  $m_i$  are monomials, and  $\deg(h_i) \leq \delta$ . The best-known blackbox (and whitebox) PIT algorithm for this model runs in quasipolynomial time [9, Corollary 6.7], and work over fields of characteristics 0 or large.

**Whitebox vs. blackbox.** Interestingly, if the input  $f$  has a whitebox access to it (for example when  $f$  is sparse, we can use [24]), then the required PIT algorithms in Theorem 1 are also in the whitebox setting. On the other hand, the factoring algorithm in [28] requires blackbox PIT algorithms. To explain it further, let  $f(x, \mathbf{z})$  be a monic polynomial (in  $x$ ) and computed by a constant-depth circuit. Further,  $f = g \cdot h$ , where  $\deg(g) \leq \delta$  and  $\gcd(g, h) = 1$ . In the usual factoring algorithm via Hensel lifting/Newton iteration, it is important to find a good starting point  $\mathbf{a} \in \mathbb{F}^n$  such that  $\gcd(g(x, \mathbf{a}), h(x, \mathbf{a})) = 1$ . This step is usually ensured by finding a hitting set for the *Resultant* polynomial  $\text{Res}_x(g(x, \mathbf{z}), h(x, \mathbf{z}))$ . Once such a point is found, one can project to the univariate  $f(x, \mathbf{a})$ , factorize it and then do the lifting. [28] observed that  $\text{Res}(g, h) = \text{Res}(g, f/g)$ . Further, they showed that the polynomials  $f/g$  as well as  $\text{Res}(g, f/g)$  can be computed by small-size constant-depth algebraic circuits. This was enough to find a good projection using [31], and then find the true factor  $g$  via lifting. Since we do not know the factor  $g$  a priori, the polynomials  $f/g$  and  $\text{Res}(g, f/g)$  can be viewed as polynomials computable by small-size constant-depth algebraic circuits *without* having explicit access to them.

### 1.3 Proof idea

In this section, we give an overview of our algorithms. The overall idea is to project the input polynomial to a trivariate polynomial, factorize it, and recover the original factors via efficient sparse interpolation [24].

**Proof ideas of Theorem 1 and Theorem 3.** Suppose  $f(x, \mathbf{z})$  is an  $(n + 1)$ -variate degree  $d$  homogeneous polynomial computed by an  $s$ -size circuit, which is *monic* in  $x$ . The monicness property can be assumed otherwise it is well-known that a random shift can make  $f$  monic, and this step can be derandomized assuming PIT for  $f$ ; see Lemma 7. We start with the simplest scenario of  $\delta = 1$ , i.e., given  $f$ , we want to output its linear factors.

Suppose  $f = \ell^e \cdot g$ , with  $e \geq 1$ , where  $\gcd(\ell, g) = 1$  and  $\ell$  is a linear polynomial. Consider the substitution  $\phi : z_i \mapsto y^i$ , where  $y$  is a new variable. Observe that  $\phi(f) \in \mathbb{F}[x, y]$  is a nonzero monic polynomial (in  $x$ ) of total degree at most  $nd$ . Further,  $\phi(\ell)$ , remains an *irreducible* factor of  $\phi(f)$  and it is easy to identify  $\ell$  from  $\phi(\ell)$ , since the monomials  $\{z_1, \dots, z_n\}$  are assigned  $y^i$  *uniquely*. One can factorize the bivariate polynomial  $\phi(f)$  in deterministic  $\text{poly}(nd)$  time (see Lemma 11). We can apply the inverse of  $\phi$  to each factor having degree 1 in  $x$ , and there could be  $nd$  many candidates of linear factors. The actual factor  $\ell$  must be one of them. The divisibility testing makes sure that it always outputs the true linear factors.

When  $\delta \geq 2$ , we can still find small weights  $w_i$ , such that any monomial  $\mathbf{z}^e$  of degree  $\leq \delta$  gets uniquely mapped to  $y^i$ , via  $\phi : z_i \mapsto y^{w_i}$ ; see Lemma 14. Unfortunately, this map may not preserve irreducibility. On the other hand, an effective version of Hilbert's Irreducibility Theorem [20] shows that a monic irreducible polynomial  $g(x, \mathbf{z})$  remains irreducible under the substitution  $z_i \mapsto \beta_i t + \gamma_i$ , where  $\beta_i, \gamma_i$  are *randomly* chosen from  $\mathbb{F}$ ; see Theorem 10. Further, this step can be derandomized when  $g$  is a low-degree polynomial. We combine these two ideas to get small weights  $w_i$  and  $w'_i$  such that the projection  $\Psi : z_i \mapsto y^{w_i} t + y^{w'_i}$  preserves the irreducibility of *any* polynomial of degree  $\leq \delta$ , and further it is *uniquely recoverable* from the projected trivariate polynomial; see Corollary 16. Therefore, it suffices to factor the trivariate polynomial, find all its irreducible factors, and recover the original factors.

The proof of Theorem 3 uses the same trivariate projection as above. In this case, we can *avoid* divisibility because the trivariate projection preserves the factorization pattern, and one can recover the original factors uniquely from the projected ones.

**Proof idea of Theorem 4.** Kaltofen and Trager [21] gave an efficient blackbox factoring algorithm, that given a blackbox access to a polynomial  $f$ , and an arbitrary point, outputs evaluations at that point of all its irreducible factors. For simplicity, consider a monic polynomial  $f(x, \mathbf{z})$ . To get the evaluations at  $(\alpha, \mathbf{c}) \in \mathbb{F}^{n+1}$ , consider a trivariate projection  $\eta : \mathbf{z} \mapsto \beta t_1 + (\mathbf{c} - \gamma)t_2 + \gamma$  and  $x \mapsto x$ , for new variables  $t_1$  and  $t_2$ . Here  $\beta, \gamma \in \mathbb{F}^n$  was chosen such that  $\mathbf{z} \mapsto \beta t + \gamma$  *preserves* the irreducibility *all* the irreducible factors of  $f$ ; such a projection exists using Theorem 10. The map  $\eta$  preserves the factorization pattern, and hence one can find the evaluations by factoring  $\eta(f)$ , and evaluating the irreducible factors at  $x = \alpha, t_1 = 0, t_2 = 1$ .

Our algorithm is a simple adaptation of their algorithm, with the following observation. Let  $g(x, \mathbf{z})$  be an irreducible sparse factor of  $f(x, \mathbf{z})$  and let  $\beta, \gamma \in \mathbb{F}^n$  be such that  $g(x, \beta t + \gamma)$  remains irreducible. Although the map  $\eta$  does not preserve the factorization pattern,  $\eta(g)$  remains an irreducible factor of  $\eta(f)$ . Therefore,  $g(\alpha, \mathbf{c})$  can be efficiently found, via evaluating the *right* trivariate factor. For finding the right factor, one can observe that there is a unique correspondence between the bivariate  $g(x, \beta t + \gamma)$  and trivariate  $\eta(g)$ . Since,  $g$  is sparse, one can use sparse interpolation [24] to explicitly reconstruct the polynomial  $g$ , from its evaluations. Finally, whether a sparse polynomial  $g$  divides the input sparse polynomial  $f$  can be solved in deterministic subexponential time, via divisibility-to-PIT reduction of [9] (see Lemma 9) and the blackbox PIT algorithm for constant-depth circuits of [31].

## 2 Preliminaries

We take  $\mathbb{F} = \mathbb{Q}$  as the underlying field throughout the paper, although the results hold over large characteristics.

Let  $\mathcal{P}(n, d)$  be the set of  $n$ -variate polynomials of degree at most  $d$ , with variables  $\mathbf{z} = (z_1, z_2, \dots, z_n)$ . For an exponent vector  $\mathbf{e} = (e_1, e_2, \dots, e_n)$ , we denote the monomial  $\mathbf{z}^{\mathbf{e}} = (z_1^{e_1}, z_2^{e_2}, \dots, z_n^{e_n})$ . Its degree is  $\|\mathbf{e}\|_1 = \sum_{i=1}^n e_i$ .

For  $\mathbf{a} \in \mathbb{F}^n$ , we also denote  $\|\mathbf{a}\|_0 = |\{i \mid a_i \neq 0\}|$ .

$\text{sp}(f)$  denotes the *sparsity*, i.e., the number of monomials with nonzero coefficients in  $f$ .

$\text{Hom}_k[f]$  denotes the homogeneous component of  $f$  of degree equal to  $k$ .

A polynomial  $f$  is called *irreducible*, if it cannot be factored into the product of two non-constant polynomials. Polynomial  $f$  is called *square-free*, if for any non-constant factor  $g$ , the polynomial  $g^2$  is not a factor of  $f$ .

By  $\deg(f)$  we denote the total degree of  $f$ . Let  $x$  and  $\mathbf{z} = (z_1, \dots, z_n)$  be variables and  $f(x, \mathbf{z})$  be a  $(n + 1)$ -variate polynomial. Then we can view  $f$  as a univariate polynomial  $f = \sum_i a_i(\mathbf{z}) x^i$  over  $\mathbb{K}[x]$ , where  $\mathbb{K} = \mathbb{F}[\mathbf{z}]$ . The  $x$ -degree of  $f$  is denoted by  $\deg_x(f)$ . It is the highest degree of  $x$  in  $f$ . Polynomial  $f$  is called *monic in  $x$* , if the coefficient  $a_{d_x}(\mathbf{z})$  is the constant 1 polynomial, i.e.  $a_{d_x}(\mathbf{z}) = 1$ , where  $d_x = \deg_x(f)$ .

An algorithm runs in *subexponential time*, if its running time on inputs of length  $n$  is bounded by  $2^{n^\epsilon}$ , for any  $\epsilon > 0$ .

## 2.1 Computational problems, complexity measures and closure properties

For classes  $\mathcal{P}, \mathcal{Q}$  of multivariate polynomials, we define the following computational problems.

- PIT( $\mathcal{P}$ ): given  $p \in \mathcal{P}$ , decide whether  $p \equiv 0$ .
- Factor( $\mathcal{P}|\mathcal{Q}$ ): given  $p \in \mathcal{P}$ , compute all its irreducible factors in  $\mathcal{Q}$  with their multiplicities.
- Div( $\mathcal{P}|\mathcal{Q}$ ): given  $p \in \mathcal{P}$  and  $q \in \mathcal{Q}$ , decide whether  $q|p$ .

The time complexity to solve these problems we denote by  $T_{\text{PIT}(\mathcal{P})}$ ,  $T_{\text{Factor}(\mathcal{P}|\mathcal{Q})}$ , and  $T_{\text{Div}(\mathcal{P}|\mathcal{Q})}$ , respectively.

► **Remark.** Note that a decision algorithm for PIT( $\mathcal{P}$ ) also yields an algorithm that computes a point  $\mathbf{a} \in (\mathbb{F} \setminus \{0\})^n$  such that  $p(\mathbf{a}) \neq 0$ , in case when  $p \not\equiv 0$ . In the blackbox case, the queries of the decision algorithm on the input of the zero-polynomial yield a *hitting set*<sup>1</sup> for the whole class  $\mathcal{P}$ . In the whitebox case, one can search for  $\mathbf{a}$  by assigning values successively to the variables and do kind of a *self-reduction*. For each variable, one tries at most  $d$  values from  $\{1, 2, \dots, d\}$  for a polynomial of degree  $d$ . If they all give 0, definitely  $d + 1$  works because it cannot be zero at  $(d + 1)$  many values. With  $n$  variables, this amounts to  $nd$  calls to the PIT-decision algorithm. The final desired point  $\mathbf{a} \in \{1, \dots, d + 1\}^n$ , which is very explicit. The running time to compute  $\mathbf{a}$  is therefore bounded by  $nd \cdot T_{\text{PIT}(\mathcal{P})}$ .

For time complexity, we assume that the polynomials are given in some model of computation, such as circuits, branching programs, or formulas. With each model, we associate a complexity measure  $\mu : \mathbb{F}[\mathbf{z}] \rightarrow \mathbb{N}$ . For example, let  $f \in \mathbb{F}[\mathbf{z}]$ , some of the commonly used measures in the literature are:

- $\mu(f) = \text{sp}(f)$ , the number of monomials with nonzero coefficients,
- $\mu(f) = \text{size}_\Delta(f)$ , the size of the *smallest* depth- $\Delta$  algebraic circuit computing  $f$ ,
- $\mu(f) = \text{size}_{\text{ROABP}}(f)$ , the *width* of the *smallest* any-order read read-once oblivious branching program (ROABP) computing  $f$ .

We define classes of polynomials of bounded measure,

$$\mathcal{C}_\mu(s, n, d) := \{f \in \mathcal{P}(n, d) \mid \mu(f) \leq s\}. \tag{1}$$

We generally assume that all polynomials we deal with can be *efficiently evaluated* at any point  $\mathbf{a} \in \mathbb{F}^n$  within the respective measure, where we consider the unit-cost model for operations over  $\mathbb{F}$ . This holds for all the computational models usually considered in the literature.

► **Definition 5 (Closure under derivatives).** *Class  $\mathcal{C}_\mu(s, n, d)$  is closed under derivatives, if for any  $f \in \mathcal{C}_\mu(s, n, d)$ , a variable  $z \in \{z_1, \dots, z_n\}$ , and  $e \in \mathbb{N}$ , the size of the derivative  $\mu(\partial^e f / \partial z^e) = \text{poly}(snd)$ , and further it can be computed in  $\text{poly}(snd)$  time from  $f$ .*

<sup>1</sup>  $H \subseteq \mathbb{F}^n$  is a hitting set for a class  $\mathcal{P}$ , if for every nonzero  $f \in \mathcal{P}$ , there exists  $\mathbf{a} \in H$ , such that  $f(\mathbf{a}) \neq 0$ .

► **Definition 6** (Closure under highest degree). Let  $f = \sum_{k=0}^d f_k(\mathbf{z}) \in \mathcal{C}_\mu(s, n, d)$ , where  $f_k(\mathbf{z}) = \text{Hom}_k[f]$ , the homogeneous component of  $f$  of degree  $k$ . We say that  $\mathcal{C}_\mu(s, n, d)$  is closed under highest degree component if in the above,  $\mu(f_d) \leq \text{poly}(snd)$ , and further it can be computed in  $\text{poly}(snd)$  time from  $f$ .

For example, if the class contains polynomials where each is a product of constant-degree polynomials, then it is *not* closed under homogenization (i.e.  $\mu(f_k) \leq \text{poly}(snd)$  and it can be computed in polynomial time). However, the highest-degree component is still a product of constant-degree polynomials. Other classes that are closed under the highest degree are sparse polynomials, polynomials computed by polynomial size any-order ROABPs, or by constant-depth circuits.

Finally, we define  $\text{Hom}[\mathcal{C}_\mu(s, n, d)]$ , as follows:

$$\text{Hom}[\mathcal{C}_\mu(s, n, d)] := \{ f \in \mathcal{P}(n, d) \mid g \in \mathcal{C}_\mu(s, n, d) \text{ and } f = \text{Hom}_d[g] \}. \quad (2)$$

When the context and the parameters are clear, we will simply denote the classes as  $\mathcal{C}$  and  $\text{Hom}[\mathcal{C}]$ , without explicitly writing the parameters.

## 2.2 Transformation to a monic polynomial

Algorithms for factoring polynomials often assume that the given polynomial is monic. If this is not the case for the given polynomial  $f$ , we apply a transformation  $\tau$  to  $f$  that yields a monic polynomial  $\tau(f)$  that we can factor. From the factors of  $\tau(f)$  we can then reveal the factors of  $f$ . Although this is standard in the literature, we state it in terms of the symbols that we introduced above.

► **Lemma 7** (Transformation to monic). Let  $\mathcal{C}$  be a class of polynomials that is closed under highest degree component. Let  $f(\mathbf{z}) \in \mathcal{C}$  be  $n$ -variate of degree  $d$  and size  $s$ . For a new variable  $x$ , and  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in (\mathbb{F} \setminus \{0\})^n$ , define a linear transformation  $\tau_\alpha$  on the variables  $z_i$ :

$$\tau_\alpha : z_i \mapsto \alpha_i x + z_i,$$

where  $\alpha_i \neq 0$ , for  $i = 1, 2, \dots, n$ . Let  $f_\alpha(x, \mathbf{z})$  be the resulting polynomial.

Then we can compute  $\boldsymbol{\alpha}$  such that  $\frac{1}{f_d(\boldsymbol{\alpha})} f_\alpha(x, \mathbf{z})$  is monic in  $x$  in time  $nd \cdot \text{T}_{\text{PIT}}(\text{Hom}[\mathcal{C}]) + \text{poly}(snd)$ , where  $f_d = \text{Hom}_d[f]$ .

**Proof.** Let  $f(\mathbf{z}) \in \mathcal{C}$  be a polynomial of degree  $d$  with  $n$  variables  $\mathbf{z} = (z_1, \dots, z_n)$ .

To see what the transformation does, let

$$f = f_0 + f_1 + \dots + f_d,$$

where  $f_k = \text{Hom}_k[f]$ , the homogeneous degree- $k$  component of  $f$ . Consider the degree- $d$  component,

$$f_d(\mathbf{z}) = \sum_{|\boldsymbol{\beta}|_1=d} c_\beta \mathbf{z}^\beta.$$

Then, for  $f_\alpha$ , we have  $\deg_x(f_\alpha) = d$  and the coefficient of the leading  $x$ -term  $x^d$  in  $f_\alpha$  is  $f_d(\boldsymbol{\alpha}) = \sum_{|\boldsymbol{\beta}|_1=d} c_\beta \boldsymbol{\alpha}^\beta$ .

Hence, the PIT algorithm for the homogeneous component  $f_d$  of  $f$  yields an  $\boldsymbol{\alpha} \in (\mathbb{F} \setminus \{0\})^n$  such that  $f_d(\boldsymbol{\alpha}) \neq 0$ . Then the polynomial  $\frac{1}{f_d(\boldsymbol{\alpha})} f_\alpha(x, \mathbf{z})$  is monic in  $x$ . For simplicity of notation, assume in the following that  $f_d(\boldsymbol{\alpha}) = 1$ , so that  $f_\alpha(x, \mathbf{z})$  is monic in  $x$ . ◀

Since we work with the shifted polynomial, we need to ensure that the shift of variables does not affect the irreducibility of the factors; this is guaranteed by the following lemma. This is quite standard in the literature; for a nice proof, see [29, Lemma B7].

► **Lemma 8.** *Let  $f(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]$  be an  $n$ -variate irreducible polynomial. Then, for every  $\mathbf{a} \in \mathbb{F}^n$ , the polynomial  $f(\mathbf{a}\mathbf{x} + \mathbf{z})$  is also irreducible.*

### 2.3 Divisibility testing reduces to PIT

Strassen [38] showed that if  $g \mid f$ , where both  $f$  and  $g$  can be computed by size  $s$  circuits, then  $h := f/g$  can also be computed by a circuit of size  $\text{poly}(sd)$ , where  $d = \deg(h)$ . Forbes [9] observed that this procedure can still be done, even when  $g \nmid f$ , and we will get a small size circuit computing a polynomial  $\tilde{h}$ . We can then argue that  $g$  divides  $f$  if and only if  $f = g \cdot \tilde{h}$ . The latter question is a PIT question.

► **Lemma 9** (Divisibility reduces to PIT, [9, Corollary 7.10]). *Let  $g(\mathbf{z})$  and  $f(\mathbf{z})$  be two polynomials of degree at most  $d$ . Let  $S \subseteq \mathbb{F}$  be a  $\text{poly}(d)$ -explicit set such that  $|S| = 2d^2 + 1$ . Further let  $\alpha \in \mathbb{F}^n$  such that  $g(\alpha) \neq 0$ . Then there are  $\text{poly}(d)$ -explicit constants  $\{c_{\beta,i}\}_{\beta \in S, 0 \leq i \leq d}$ , such that*

$$g(\mathbf{z}) \mid f(\mathbf{z}) \iff f(\mathbf{z} + \alpha) - g(\mathbf{z} + \alpha) \cdot \sum_{\beta \in S} f(\beta\mathbf{z} + \alpha) \sum_{0 \leq i \leq d} c_{\beta,i} \cdot g(\beta\mathbf{z} + \alpha)^i = 0$$

### 2.4 Effective Hilbert's Irreducibility Theorem

An effective version of Hilbert's Irreducibility Theorem due to Kaltofen and von zur Gathen shows how to project a multivariate irreducible polynomial down to two variables, such that the projected bivariate polynomial stays irreducible. The proof shows the existence of an *irreducibility certifying polynomial*  $G(\mathbf{a}, \mathbf{b})$  in  $2n$  variables corresponding to the irreducible polynomial  $g(x, \mathbf{z})$ . The nonzeroness of  $G$  proves the irreducibility of  $g(x, \mathbf{z})$  and also gives a way to find an irreducibility-preserving projection to bivariate (see [17, 20, 27]).

► **Theorem 10.** *Let  $g(x, \mathbf{z})$  be an irreducible polynomial of total degree  $\delta$  with  $n + 1$  variables that is monic in  $x$ . There exists a nonzero polynomial  $G(\mathbf{a}, \mathbf{b})$  of degree  $2\delta^5$  in  $2n$  variables such that for  $\alpha, \beta \in \mathbb{F}^n$ ,*

$$G(\alpha, \beta) \neq 0 \implies \widehat{g}(x, t) = g(x, \alpha_1 t + \beta_1, \dots, \alpha_n t + \beta_n) \text{ is irreducible.}$$

The certifying polynomial  $G$  immediately yields a randomized algorithm to construct the irreducible projection  $\widehat{g}$  via PIT. The derandomization of Hilbert's Irreducibility Theorem is a challenging open problem in general. We observe that it can be derandomized for constant degree polynomials.

### 2.5 Basics of factoring and interpolation

Berlekamp [2] and Lenstra, Lenstra and Lovász [30] gave efficient factorization algorithms for *univariate* polynomials over finite fields and  $\mathbb{Q}$ , respectively. Kaltofen [18] showed how to reduce the factorization of *bivariate* polynomials to univariate polynomials. In fact, the reduction works for  $k$ -variate polynomials, for any constant  $k$ . In our case, we use it for the case  $k = 3$ .

Via standard interpolation, one can assume that the input is given as a dense representation.

## 75:10 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

► **Lemma 11** (Trivariate Factorization). *Let  $f(x, y, z)$  be a trivariate polynomial of degree  $d$ . Then there exists an algorithm that outputs all its irreducible factors and their multiplicities in time  $\text{poly}(d)$ .*

The following lemma shows how to find the multiplicity of an irreducible factor  $g$  of a polynomial  $f$ . It holds when  $\text{char}(\mathbb{F}) = 0$ , or, large. For a concise proof, see [28, Lemma 4.1].

► **Lemma 12** (Factor multiplicity). *Let  $f(\mathbf{z}), g(\mathbf{z}) \in \mathbb{F}[\mathbf{z}]$  be non-zero polynomials and let  $z \in \{z_1, \dots, z_n\}$  be such that  $\partial_z(g) \neq 0$  and  $g$  is irreducible. Then the multiplicity of  $g$  in  $f$  is the smallest non-negative integer  $e$  such that  $g \nmid \frac{\partial^e f}{\partial z^e}$ .*

Klivans and Spielman [24] derandomized the isolation lemma for PIT of sparse polynomials. Their algorithm works over fields of 0 or large characteristics.

► **Lemma 13** (Sparse PIT and interpolation). *Given an  $n$ -variate  $s$ -sparse polynomial  $f$  of degree  $d$  via blackbox access, PIT for  $f$  works in time  $\text{poly}(snd)$ . Furthermore, if  $f$  is nonzero, one can find the monomials in  $f$  with nonzero coefficients in time  $\text{poly}(snd)$ .*

► **Remark.** Let  $\mathbb{F} = \mathbb{Q}$ , for the simplicity. The interpolation algorithm in [24] works by projecting  $f$  to a univariate polynomial in  $y$  via the map  $z_i \mapsto p_i y^{w_i}$ , for  $p_i$  are distinct primes, and weights  $w_i$ . They used univariate interpolation, to find the coefficients and the exponents. If the input polynomial  $f$  is not  $s$ -sparse, one can still run the algorithm. If at any moment while doing the univariate interpolation, it detects more than  $s$  many nonzero coefficients, it stops, otherwise it will continue, and indeed at the end, output a wrong  $s$ -sparse polynomial  $\hat{f}$ , such that (unfortunately) it matches at all the interpolating values. Given  $s, n, d$ , the evaluation points on which the interpolation algorithm can be thought as coming from a fixed set.

### 2.6 Isolation

Let  $M_\delta$  be the set of monomials in  $n$  variables  $\mathbf{z} = (z_1, z_2, \dots, z_n)$  of degree bounded by  $\delta$ ,

$$M_\delta = \{ \mathbf{z}^e \mid \|e\|_1 \leq \delta \}. \quad (3)$$

Note that  $M_\delta$  is polynomially bounded, for constant  $\delta$ ,

$$|M_\delta| \leq \binom{n + \delta}{\delta} \leq (n + \delta)^\delta \leq (\delta + 1) n^\delta = O(n^\delta). \quad (4)$$

There is a standard way to map the multivariate monomials in  $M_\delta$  in a injective way to univariate monomials of polynomial degree. For completeness, we describe the details.

Consider the standard *Kronecker substitution* on  $M_\delta$ . Define

$$\varphi : z_i \mapsto y^{(\delta+1)^{i-1}}. \quad (5)$$

By extending  $\varphi$  linearly to monomials  $\mathbf{z}^e \in M_\delta$ , we get

$$\varphi : \mathbf{z}^e \mapsto y^{\sum_{i=1}^n e_i (\delta+1)^{i-1}}, \quad (6)$$

Clearly,  $\varphi$  is injective on  $M_\delta$ . However, the degree of  $y$  can be exponentially large, up to  $(\delta + 1)^n$ . A way around is to take the exponents modulo some small prime number  $p$ . We have to determine  $p$  in a way to keep the mapping injective on  $M_\delta$ . Hence, for any two terms  $y^e, y^{e'}$  we get from  $\varphi$ , we have to ensure that  $e \not\equiv e' \pmod{p}$ . Equivalently  $p \nmid (e - e')$ .

We have  $|e - e'| \leq (\delta + 1)^n$  and, by (4), there are  $(\delta + 1)^2 n^{2\delta}$  many pairs  $e, e'$  we get from  $M_\delta$  via  $\varphi$ . Prime  $p$  should not divide any of these differences, and hence,  $p$  should not divide their product  $P$ . The product  $P$  is bounded by

$$P \leq ((\delta + 1)^n)^{(\delta+1)^2 n^{2\delta}} = (\delta + 1)^{(\delta+1)^2 n^{2\delta+1}}. \quad (7)$$

Hence,  $P$  has at most  $\log P \leq R = (\delta + 1)^3 n^{2\delta+1}$  many prime factors. By the Prime Number Theorem, there are more than  $\log P$  primes in the set  $[R^2]$ . Hence, we can find an appropriate prime  $p \leq R^2 = n^{O(\delta)}$ .

► **Lemma 14.** *There is a prime  $p = n^{O(\delta)}$  such that the linear extension of*

$$\varphi_p : z_i \mapsto y^{w_i}, \quad \text{where } w_i = (\delta + 1)^{i-1} \bmod p, \quad \text{for } i = 1, 2, \dots, n, \quad (8)$$

*to monomials is injective on  $M_\delta$ . Moreover, we can find such a  $p$  in time  $n^{O(\delta)}$  and compute and invert  $\varphi_p$  in time  $n^{O(\delta)}$ .*

**Proof.** We already argued about the existence of prime  $p$ . For the running time, recall that  $|M_\delta| = O(n^\delta)$ . Therefore we can search for  $p$  and check whether it works on  $M_\delta$  in time  $n^{O(\delta)}$ . At the same time we can compute pairs of exponents  $(e, k)$  such that  $\varphi_p(z^e) = y^k$ . These pairs can be used to invert  $\varphi_p$ . ◀

The mapping  $\varphi_p$  in Lemma 14 maintains factors of degree  $\delta$  of a polynomial in the following sense.

► **Lemma 15.** *Let polynomial  $f(\mathbf{z})$  factor as  $f = gh$ , where  $g(\mathbf{z})$  has degree  $\delta$ . Let  $\varphi_p$  be the map from Lemma 14. Then we have  $\varphi_p(f) = \varphi_p(g)\varphi_p(h)$ , and  $g$  can be recovered from  $\varphi_p(g)$  in time  $n^{O(\delta)}$ .*

Note that in Lemma 15, we do *not* claim that irreducibility is maintained: when  $g$  is irreducible, still  $\varphi_p(g)$  might be reducible. Consider the example  $n = \delta = 2$ . The weights  $\{1, 3\}$  make sure that each monomial  $z_1^2, z_1 z_2, z_2^2$  gets mapped to a distinct power in  $y$ . Let  $g(x, z) = x^2 - z_1 z_2$ . Observe that  $g$  is irreducible, however  $g(x, y, y^3) = (x - y^2)(x + y^2)$  is reducible.

We combine Lemma 14 and Theorem 10 to obtain a projection of a multivariate polynomial to a 3-variate polynomial that maintains irreducibility of polynomials up to degree  $\delta$ .

► **Corollary 16.** *Let  $g(x, \mathbf{z})$  be an irreducible polynomial of constant degree  $\delta$  with  $n + 1$  variables that is monic in  $x$ . There exists  $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$  with  $w_i, w'_i = n^{\text{poly}(\delta)}$  such that*

$$\Psi(g) = g(x, y^{w_1}t + y^{w'_1}, \dots, y^{w_n}t + y^{w'_n}) \in \mathbb{F}[x, y, t] \quad (9)$$

*is irreducible. Moreover, we can compute and invert  $\Psi(g)$  in time  $n^{\text{poly}(\delta)}$ .*

**Proof.** Let  $G(\mathbf{a}, \mathbf{b})$  be the polynomial of degree  $2\delta^5$  in  $2n$  variables provided by Theorem 10 for  $g$ . Let  $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$  with  $w_i, w'_i = n^{\text{poly}(\delta)}$  be the exponents we get from Lemma 14 for  $G$ . That is,

$$\widehat{G}(y) = G(y^{w_1}, \dots, y^{w_n}, y^{w'_1}, \dots, y^{w'_n}) \neq 0.$$

Now, suppose that  $\Psi(g)$  is reducible. Then it would also be reducible at a point  $y = \alpha$ , where  $\widehat{G}(\alpha) \neq 0$ . But then  $\widehat{g}(x, t) = \Psi(g)(x, \alpha, t)$  would be reducible too, and this would contradict Theorem 10. We conclude that  $\Psi(g)$  is irreducible.

## 75:12 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

For the complexity, we first determine prime  $p$  from Lemma 14 and then get the weights  $\mathbf{w}, \mathbf{w}'$  from above. For a given  $g(x, \mathbf{z}) = \sum_{k, \mathbf{e}} c_{k, \mathbf{e}} x^k \mathbf{z}^{\mathbf{e}}$ , we can compute  $\Psi(g)$  in time  $n^{\text{poly}(\delta)}$ . For a monomial of  $g$ , the mapping looks as follows:

$$c_{k, \mathbf{e}} x^k \mathbf{z}^{\mathbf{e}} \mapsto c_{k, \mathbf{e}} x^k \prod_{i=1}^n (y^{w_i} t + y^{w'_i})^{e_i}. \quad (10)$$

To compute  $g$  from  $\Psi(g)$ , set  $t = 0$ , i.e. consider  $\Psi(g)(x, y, 0)$ . From (10) we see that monomials then have the form

$$c_{k, \mathbf{e}} x^k y^{\sum_{i=1}^n e_i w'_i}.$$

From these we get the exponents  $k$  and  $\mathbf{e}$  similar as in the proof of Lemma 14.  $\blacktriangleleft$

► **Remark.** In Corollary 16, when we say that we *invert*  $\Psi$ , it means that for a given  $h \in \mathbb{F}[x, y, t]$  which is monic in  $x$  with  $x$ -degree  $\leq \delta$ , we either detect that  $h$  is not in the codomain of  $\Psi$ , or we compute  $g \in \mathbb{F}[x, \mathbf{z}]$  such that  $\Psi(g) = h$  in time  $n^{\text{poly}(\delta)}$ .

The inversion can be done similarly as described in the proof of Corollary 16. One can evaluate  $t = 0$ , and then for every monomial  $x^k y^j$ , try to find  $x^k \mathbf{z}^{\mathbf{e}}$  that would map to such a monomial at  $t = 0$ . By the property of the map, while mapping the  $y$ -degrees,  $\mathbf{z}$ -degree could be at most  $\delta$ , i.e.  $\deg(x^k \mathbf{z}^{\mathbf{e}}) \leq \delta$ . We will, of course, return empty if the degree of any such monomial, after inverting, becomes  $> \delta$ . Finally, once we have got a candidate  $g$  of degree  $\delta$ , we still have to check whether  $\Psi(g) = h$ , because the inversion procedure ignores the variable  $t$ . The last step can also be efficiently checked.

The polynomial  $g$  of degree  $\delta$  we considered so far can be thought to be a constant-degree factor of a given polynomial  $f$  of degree  $d$ . Our goal would be to compute  $g$ . It is now easy to extend the above results to hold for all degree- $\delta$  factors of  $f$  simultaneously.

► **Corollary 17.** *Let  $f(x, \mathbf{z})$  be a polynomial of degree  $d$  with  $n + 1$  variables that is monic in  $x$ , and let  $\delta$  be a constant. There exists  $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$  with  $w_i, w'_i = dn^{\text{poly}(\delta)}$  such that for any irreducible factor  $g$  of degree  $\delta$  of  $f$ , we have that  $\Psi(g)$  is an irreducible factor of  $\Psi(f)$ .*

**Proof.** The proof goes along the lines of Corollary 16, but we choose the weights slightly larger so that the  $\widehat{G}(y)$  polynomials for *all* the degree- $\delta$  factors  $g$  of  $f$  are non-zero simultaneously. That is, we choose prime  $p$  in Lemma 14 as  $p = dn^{\text{poly}(\delta)}$ .  $\blacktriangleleft$

Finally, we conclude this subsection by a general remark that whenever  $n$  and  $\delta$  are fixed, these weights are fixed and can be found efficiently.

### 3 Computing the low-degree factors

For a size measure  $\mu$ , we consider a class of polynomials  $\mathcal{C} = \mathcal{C}_\mu(s, n, d) \subseteq \mathcal{P}(n, d)$  such that  $\mathcal{C}$  is closed under derivatives. Many classes  $\mathcal{C}$  in the literature fulfill this condition. Useful for us are in particular sparse polynomials, polynomials computed by poly-size any-order ROABPs or by constant-depth circuits. For a constant  $\delta \in \mathbb{N}$ , let  $\mathcal{D} = \mathcal{P}(n, \delta)$ .

Our first theorem shows that for any polynomial  $f \in \mathcal{C}$ , all the factors of  $f$  that are in  $\mathcal{D}$  can be computed in polynomial time with oracles for PIT for  $\text{Hom}[\mathcal{C}]$  and divisibility testing  $\mathcal{C}$  by  $\mathcal{D}$ .

► **Theorem 18.** *Factor( $\mathcal{C}|\mathcal{D}$ ) can be solved deterministically in time*

$$nd \cdot \text{T}_{\text{PIT}(\text{Hom}[\mathcal{C}])} + d^2 n^{\text{poly}(\delta)} \cdot \text{T}_{\text{Div}(\mathcal{C}/\mathcal{D})} + \text{poly}(s, n^{\text{poly}(\delta)}, d).$$

**Proof.** Let  $f(\mathbf{z}) \in \mathcal{C}$ . To compute the factors of degree  $\delta$  of  $f$ , we first do some transformations. The first step is to make  $f$  monic in a new variable  $x$  via Lemma 7. Let  $f_\alpha(x, \mathbf{z})$  be monic.



Then we apply Corollary 17 to  $f_\alpha(x, z)$ . That is we compute the weights  $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$  bounded by  $dn^{\text{poly}(\delta)}$  and explicitly compute  $\Psi(f_\alpha) \in \mathbb{F}[x, y, t]$  of degree at most  $\tilde{d} = d^2 n^{\text{poly}(\delta)}$ . Note that the  $x$ -degree of  $f_\alpha$  has not changed by mapping  $\Psi$ . In the blackbox case, we can interpolate and reconstruct the polynomial in time  $\text{poly}(s\tilde{d})$ .

The next step is to factor 3-variate  $\Psi(f_\alpha)$ . This can be done efficiently Lemma 11. Finding and listing all the irreducible factors takes time  $\text{poly}(\tilde{d})$ .

Having the factors of  $\Psi(f_\alpha)$  in hand, we invert transformations  $\Psi$  and  $\tau_\alpha$  on the factors. Let  $\tilde{g}$  be a factor of  $\Psi(f_\alpha)$ . By Corollary 17, if  $g$  indeed corresponds to a  $\delta$ -degree factor of  $f$ , then  $\tau_\alpha(g)$  corresponds to a  $\delta$ -degree factor of  $\tau_\alpha(f)$ . Therefore, the inverse transformations will yield  $g$ . Formally, the factor is  $g = \tau_\alpha^{-1}(\Psi^{-1}(\tilde{g}))$ .

However  $\tilde{g}$  might also *not* correspond to a degree- $\delta$  factor of  $f$ . In this case, either the inverse transformation does not go through properly, or the degree we get is larger than  $\delta$ . In these cases, we can immediately throw away  $\tilde{g}$ ; see the remark after Corollary 16. But it could also be that we actually obtain a polynomial  $g$  of degree  $\delta$ , just that it is not a factor of  $f$ . For that reason, we finally do a divisibility check whether  $g|f$ . That way we will compute all factors of  $f$  of degree  $\delta$ .

For the time complexity of the factoring algorithm, we have  $nd \cdot \text{T}_{\text{PIT}(\text{Hom}[\mathcal{C}]})$  for transforming  $f$  to monic  $f_\alpha$  by Lemma 7. Time  $\text{poly}(dn^{\text{poly}(\delta)})$  is used for map  $\Psi$  and the factoring of  $\Psi(f_\alpha)$ . Similar time is taken to invert and get the candidate factors. Finally, we have at most  $d^2 n^{\text{poly}(\delta)}$  candidate polynomials  $g$  for which we test divisibility of  $g|f$ . ◀

► **Remark.** Theorem 18 can be applied in different algebraic models. Furthermore, if a class is closed under highest degree, one can simply assume PIT for  $\mathcal{C}$ . In particular, if we work with algebraic formulas, or algebraic branching programs (ABPs), then the above theorem along with the divisibility lemma Lemma 9 implies that we need PIT for the *same* class, to deterministically find the constant-degree factors.

The following pseudo-code summarizes the algorithm given in the proof of Theorem 18.

■ **Algorithm 1** Computing factors of degree  $\leq \delta$ .

---

**Input** :  $f(z)$ ,  $s$ , and  $\delta$ , where  $f$  is an  $n$ -variate polynomial of degree  $d$  such that  $\mu(f) \leq s$ , and  $\delta$  is a constant.

**Output** : A list of irreducible polynomials of degree  $\leq \delta$ , which are factors of  $f$ , along with the factor-multiplicities.

- 1 Set the output list  $L = \emptyset$ . Set the intermediate candidates list  $L' = \emptyset$ .
- 2 Make a monic transformation  $\tau_\alpha : z_i \mapsto \alpha_i x + z_i$ , according to Lemma 7. Let  $f_\alpha(x, z) := \tau_\alpha(f)$ .
- 3 Find weights  $\mathbf{w}, \mathbf{w}'$  bounded by  $dn^{\text{poly}(\delta)}$  according to Corollary 17 and compute  $\Psi(f_\alpha) \in \mathbb{F}[x, y]$  in dense representation.
- 4 Factorize the trivariate polynomial  $\Psi(f_\alpha)$  according to Lemma 11. Let  $S$  be the set of all  $\leq \delta$  degree factors in  $x$  of  $\Psi(f_\alpha)$  in dense representation.
- 5 **for**  $\tilde{g} \in S$  **do**
  - 6     /\* Computing candidate factors via divisibility \*/
  - 7     Compute  $\hat{g} = \Psi^{-1}(\tilde{g})$  (if the inverse exists) of degree  $\leq \delta$  by Corollary 16.
  - 8     Compute  $g = \tau_\alpha^{-1}(\hat{g})$ .
  - 9     If  $g|f$  then add  $g$  to  $L'$ .
- 9 **for**  $g \in L'$  **do**
  - 10    /\* Computing multiplicities via Lemma 12 \*/
  - 11    Let  $z \in \{z_1, \dots, z_n\}$  be any variable that  $g$  depends on, so that  $\partial_z(g) \neq 0$ .
  - 12    Find the smallest  $e \geq 0$  such that  $g \nmid \frac{\partial^e f}{\partial z^e}$ .
  - 13    **if**  $e > 1$  **then** add  $(g, e)$  to the list  $L$ .
- 13 **return**  $L$

---

#### 4 Computing the factors of a constant degree product

Now, we want to output the constant-degree irreducible factors in the promise case.

It is still an open question to test in deterministic polynomial time if a quadratic polynomial (or any non-linear polynomial whose total degree is upper bounded by a constant) divides another polynomial  $f$ , even when  $f$  is a sparse polynomial. In the above theorem  $f$  may not be sparse and we have only blackbox access to it. Thus, the proof for Theorem 2 is a bit different from Theorem 1.

► **Theorem 19.** *Given blackbox access to an  $n$ -variate degree- $d$  polynomial  $f = \prod_{i=1}^s g_i^{e_i}$ , where  $g_i$  are irreducible polynomials with  $\deg(g_i) \leq \delta$ , one can deterministically output all  $(g_i, e_i)$  in time  $\text{poly}(dn^{\text{poly}(\delta)})$ .*

**Proof.** The first step of the algorithm is to make  $f(\mathbf{z})$  monic in a new variable  $x$  via Lemma 7. One can find an  $\alpha \in (\mathbb{F} \setminus \{0\})^n$  in  $\text{poly}(d, n^\delta)$  time by using the hitting set for polynomials of degree  $\leq \delta$ ; see [7, 5].

Next step is to apply Corollary 16 to  $f_\alpha(x, \mathbf{z})$ . Find the weights  $\mathbf{w}, \mathbf{w}' \in \mathbb{F}^n$ , each bounded by  $dn^{\text{poly}(\delta)}$ . Observe that  $\Psi(f_\alpha) \in \mathbb{F}[x, y, t]$ , of degree at most  $\tilde{d} := d^2 n^{\text{poly}(\delta)}$ . By the same lemma, we know that for any irreducible factor  $g$  of  $f$ , we have that  $\Psi(g)$  is an irreducible factor of  $\Psi(f)$ .

The next step is to explicitly compute the trivariate polynomial  $\Psi(f_\alpha)$ . Since, the degree of the polynomial is at most  $\tilde{d}$ , one can interpolate and reconstruct the polynomial, from its blackbox access, in time  $\text{poly}(s\tilde{d})$ . Note that  $s \leq d$ .

The next step is to factorize  $\Psi(f_\alpha)$ . This can be done efficiently using Lemma 11. Finding and listing all the irreducible factors takes time  $\text{poly}(\tilde{d})$ .

The next step is to invert the transformation  $\Psi^{-1}$  on the factors computed in the previous step. This can be done efficiently in time  $\text{poly}(dn^{\text{poly}(\delta)})$ ; see Corollary 16 and its remark. Let  $\tilde{g}$  be a factor of  $\varphi(f_\alpha)$ . Output  $g = \tau_\alpha^{-1}(\Psi^{-1}(\tilde{g}))$ .

■ **Algorithm 2** Promise factors of degree  $\leq \delta$ .

---

**Input** : An  $n$ -variate, degree  $d$  polynomial  $f(\mathbf{z})$ , and a constant  $\delta$ , and a promise that all its irreducible factors have degree  $\leq \delta$ .

**Output** : All the irreducible factors of  $f$ , along with the multiplicities.

- 1 Set the output list  $L = \emptyset$ .
  - 2 Make a monic transformation  $\tau_\alpha : z_i \mapsto \alpha_i x + z_i$ , according to Lemma 7. Let  $f_\alpha(x, \mathbf{z}) := \tau_\alpha(f)$ .
  - 3 Find weights  $\mathbf{w}, \mathbf{w}'$  bounded by  $dn^{\text{poly}(\delta)}$  according to Corollary 17 and compute  $\Psi(f_\alpha) \in \mathbb{F}[x, y]$  in dense representation.
  - 4 Factorize the trivariate polynomial  $\Psi(f_\alpha)$  according to Lemma 11. Let  $S$  be the set of irreducible factors of  $\Psi(f_\alpha)$  along with its multiplicities as a tuple.
  - 5 **for**  $(\tilde{g}, e) \in S$  **do**
    - 6 /\* Computing the irreducible factors via inversion \*/
    - 6 Compute  $\hat{g} = \Psi^{-1}(\tilde{g})$ , by Corollary 16 and its remark.
    - 7 Compute  $g = \tau_\alpha^{-1}(\hat{g})$ , and add  $(g, e)$  to  $L$ .
  - 8 **return**  $L$
-

Now, we discuss the correctness of the output. By assumption,  $f = g_1^{e_1} \cdots g_s^{e_s}$  where  $g_i \in \mathcal{D}$ . Therefore,  $\Psi(\tau_\alpha(f)) = \Psi(\tau_\alpha(g_1))^{e_1} \cdots \Psi(\tau_\alpha(g_s))^{e_s}$ . Furthermore, by choice of  $\mathbf{w}, \mathbf{w}'$ , we know the following facts.

1. The polynomials  $\Psi(\tau_\alpha(g_i)) \in \mathbb{F}[x, y, t]$  are irreducible over  $\mathbb{F}$ , for all  $i = 1, \dots, s$ .
2.  $\Psi(\tau_\alpha(g_i)) \neq \Psi(\tau_\alpha(g_j))$ , for  $i \neq j$ .
3. One can uniquely recover  $g_i$  from  $\Psi(\tau_\alpha(g_i))$ , using Corollary 16 and its remark.

This implies that the factoring pattern of  $f$  and  $\Psi(\tau_\alpha(f))$  remain the same, and can be recovered by simply looking at the factorization of  $\Psi(\tau_\alpha(f))$ .

**Time complexity.** To make  $f$  monic, we find vector  $\alpha$  in time  $\text{poly}(dn^\delta)$ . Interpolation and trivariate factorization of a degree- $\tilde{d}$  polynomial takes time  $\text{poly}(\tilde{d}) = \text{poly}(dn^{\text{poly}(\delta)})$ , see Lemma 11. Inverting each of them to get the original factor also takes time  $\text{poly}(dn^{\text{poly}(\delta)})$ .  $\blacktriangleleft$

## 5 Finding sparse factors reduces to sparse irreducibility

Let  $f$  be an  $n$ -variate irreducible polynomial of degree  $d$ . Let  $\alpha \in (\mathbb{F} \setminus \{0\})^n$ , such that

$$f_d(\alpha) \neq 0, \text{ where } f_d = \text{Hom}_d[f] \text{ is the homogeneous degree } -d \text{ component of } f.$$

Using Lemma 7 and Lemma 8, one can conclude that  $\hat{f}(x, \mathbf{z}) := f(\alpha x + \mathbf{z})$  is a monic irreducible  $(n+1)$ -variate polynomial of degree  $d$ .

On the other hand Theorem 10 shows that for a *random*  $\beta, \gamma \in \mathbb{F}^n$ , the bivariate polynomial

$$f(\alpha x + \beta t + \gamma) = \hat{f}(x, \beta t + \gamma) \in \mathbb{F}[x, t],$$

remains irreducible. When the degree of  $f$  is a constant, such an irreducibility preserving reduction can be efficiently derandomized; see Corollary 16. Formally, we should think of it as a derandomization for the class of constant degree polynomials. In Corollary 16, one can think of evaluating  $y$  at polynomially many points to find the right  $\beta, \gamma$ , while the point  $\alpha$  comes from an efficient PIT algorithm for  $f_d$ . Note that *any*  $\alpha \in (\mathbb{F} \setminus \{0\})^n$  suffices as long as  $f_d(\alpha) \neq 0$ .

Let  $\alpha \in (\mathbb{F} \setminus \{0\})^n$ . We define a set  $S_\alpha(s, n, d)$  as follows.

$$S_\alpha(s, n, d) := \{f \in \mathcal{P}(n, d) \mid f \text{ is irreducible, } \text{sp}(f) \leq s, \text{ and } \text{Hom}_{\text{deg}(f)}[f](\alpha) \neq 0\}. \quad (11)$$

Motivated by the efficient derandomization of irreducibility preserving bivariate projections for constant degree polynomials, we assume the following.

► **Assumption 1 (Sparse irreducible projection).** Let  $\alpha \in (\mathbb{F} \setminus \{0\})^n$ , and  $S_\alpha(s, n, d) \subseteq \mathcal{P}(n, d)$  as defined in Equation (11). Then, there is a deterministic subexponential time algorithm to find an explicit set  $\mathcal{H}_\alpha \subseteq \mathbb{F}^{2n}$  of size subexponential, such that for any  $f \in S_\alpha(s, n, d)$ , there exists  $(\beta, \gamma) \in \mathcal{H}_\alpha$  such that  $f(\alpha x + \beta t + \gamma)$  remains irreducible.

► **Remark.** Assuming the above, one can decide whether an  $s$ -sparse degree- $d$  polynomial  $f$  is irreducible or not in subexponential time. To do this, one can find  $\alpha \in (\mathbb{F} \setminus \{0\})^n$  such that  $\text{Hom}_d[f](\alpha) \neq 0$ , in time  $\text{poly}(snd)$ , using [24]. Hence, one can find a set  $\mathcal{H}_\alpha$  such that  $f(\mathbf{z})$  is reducible if and only if  $f(\alpha x + \beta t + \gamma)$  is reducible, for every  $(\beta, \gamma) \in \mathcal{H}_\alpha$ . Whether a bivariate polynomial is reducible can be checked in time  $\text{poly}(d)$  (Lemma 11).

## 75:16 Derandomizing Multivariate Polynomial Factoring for Low Degree Factors

Using Assumption 1, one can design an efficient deterministic algorithm to output sparse irreducible factors of a sparse polynomial.

► **Theorem 20** (Efficient sparse factoring). *Let  $f \in \mathbb{F}[\mathbf{z}]$  be an  $s$ -sparse polynomial of degree  $d$ . If Assumption 1 holds, then there is a deterministic subexponential time algorithm that outputs all its irreducible factors with sparsities  $\leq s$ , along with the multiplicities.*

**Proof.** Assume that  $g$  is a  $s$ -sparse irreducible factor of  $f$  with multiplicity  $e$ , i.e.,  $f = g^e \cdot R$ , where  $\gcd(g, R) = 1$ . Assume that  $\deg(g) = d_1$ , and  $\deg(R) = d_2$ . We will argue that Algorithm 3 correctly outputs  $(g, e)$ . Let  $\alpha \in (\mathbb{F} \setminus \{0\})^n$ , such that  $\text{Hom}_d[f](\alpha) \neq 0$ . Observe that  $\text{Hom}_d[f] = (\text{Hom}_{d_1}[g])^e \cdot \text{Hom}_{d_2}[R]$ . Therefore, it must hold that  $\text{Hom}_{d_1}[g](\alpha) \neq 0$ , implying  $g \in S_\alpha(s, n, d)$ .

By Assumption 1, we know that there exists a subexponential time algorithm to find a set  $\mathcal{H}_\alpha$  such that  $g(\alpha x + \beta t + \gamma)$  remains irreducible for some  $(\beta, \gamma) \in \mathbb{F}^{2n}$ . We call  $(\beta, \gamma)$  a “good” point for  $g$ .

For such a good point, the set  $S$  in Line 6 of Algorithm 3 must contain  $g(\alpha x + \beta t + \gamma)$ . Pick any  $\mathbf{c} \in \mathbb{F}^n$ . Observe that  $\tilde{g}(x, t_1, t_2) := \phi_{\mathbf{c}}(g) = g(\alpha x + \beta t_1 + (\mathbf{c} - \gamma)t_2 + \gamma)$  is a monic polynomial in  $x$ . Further,  $\tilde{g}$  remains irreducible, since  $\tilde{g}(x, t, 0) = g(\alpha x + \beta t + \gamma)$  is irreducible by the choice of a good point  $(\beta, \gamma)$ . Hence,  $S'$  in Line 10 must contain the polynomial  $\tilde{g}$ .

One can find the corresponding factor in Line 12, and suppose the corresponding index is  $j$ . Note that  $\tilde{g}(0, 0, 1) = g(\mathbf{c})$ .

From the above, one can conclude that the  $L'_j$  contains  $(j, \mathbf{c}, g(\mathbf{c}))$ . One can now do the sparse interpolation using [24], to reconstruct  $g$ ; see Lemma 13. Since  $g \mid f$ , this is correctly detected and added to the list  $L'$ . This is being discussed in Line 15-16.

Further, by our choice of a good point  $(\beta, \gamma) \in \mathcal{H}_\alpha$ , the bivariate  $g(\alpha x + \beta t + \gamma)$  remains irreducible, and hence it successfully passes Line 19. Since  $g$  is a nontrivial polynomial, one can find a variable  $z \in \{z_1, \dots, z_n\}$  such that  $\partial_z(g) \neq 0$  in Line 20. Using Lemma 12, one can conclude that indeed Line 22 adds  $(g, e)$  to the list  $L$ .

From the above analysis, we know that Algorithm 3 always outputs  $g$  with the corresponding multiplicity. On the other hand, Line 15 makes sure that the candidate polynomial is indeed at most  $s$ -sparse (see Lemma 13) and Line 16, by the divisibility testing, makes sure it detects a wrong sparse factor. What could have happened is  $L'$  contains factors which are  $s$ -sparse reducible polynomials dividing  $f$ .

Line 19 again uses Assumption 1 to check if it is indeed irreducible or not. Finally, Line 20-22 make sure that the Algorithm 3 never outputs the correct multiplicity. This finishes the correctness of the algorithm.

**Running time analysis.** Since  $f$  is  $s$ -sparse, one can find an  $\alpha \in \mathbb{F}^n$ , such that  $\text{Hom}_d[f](\alpha) \neq 0$ , in  $\text{poly}(snd)$  time [24].

Line 6-10 take  $\text{poly}(d)$  time, since bivariate/trivariate interpolation and factorization can be done efficiently Lemma 11.

Line 15 can be done using the sparse interpolation algorithm in  $\text{poly}(snd)$  time [24]. Line 16 requires whether a given  $s$ -sparse polynomial  $P_j$  divides another  $s$ -sparse polynomial  $f$  or not. Using the techniques from [9] (Lemma 9), this divisibility question can be reduced to a PIT instance of a constant-depth circuit, for which there is a subexponential time algorithm [31].

Line 19 is again bivariate factorization Lemma 11 which can be done in  $\text{poly}(d)$  time.

Further,  $\frac{\partial^e f}{\partial z^e}$  is at most  $s$ -sparse. Hence, the divisibility question of whether  $P \mid \frac{\partial^e f}{\partial z^e}$  in Line 21, can be similarly done in subexponential time. Finally, since  $|\mathcal{H}_\alpha|$  is subexponentially large, the overall running time remains subexponential. ◀

■ **Algorithm 3** Computing  $s$ -sparse factors.

**Input** : An  $n$ -variate, degree  $d$  polynomial  $s$ -sparse polynomial  $f(\mathbf{z})$ .

**Output** : A list of irreducible  $s$ -sparse polynomials which are factors of  $f$ , along with the factor-multiplicities.

```

1 Set the output list  $L = \emptyset$ . Set the intermediate candidate list  $L' = \emptyset$ .
2 Use Lemma 13 to find an  $\alpha \in (\mathbb{F} \setminus \{0\})^n$  such that  $\text{Hom}_d[f](\alpha) \neq 0$ .
3 Use Assumption 1 to find a set  $\mathcal{H}_\alpha$ .
4 for each  $(\beta, \gamma) \in \mathcal{H}_\alpha$  do
5   Let  $\phi : z_i \mapsto \alpha_i x + \beta_i t + \gamma_i$ . Compute  $\hat{f} := \phi(f) \in \mathbb{F}[x, t]$ , as a dense
   representation.
6   Factorize the bivariate polynomial  $\hat{f}$  over  $\mathbb{F}$ . Let  $S = \{g_1(x, t), \dots, g_m(x, t)\}$  be
   the set of its irreducible factors.
7   Set  $m$  many interpolating lists  $L'_j = \emptyset$ , for  $j \in [m]$ .
8   Fix  $\mathbf{c} = (c_1, \dots, c_n) \in \mathbb{F}^n$ . /* These points are the evaluation points
   for which  $s$ -sparse interpolation succeeds */
9   For new variables  $t_1$  and  $t_2$ , define a new map
    $\phi_{\mathbf{c}} : z_i \mapsto \alpha_i x + \beta_i t_1 + (c_i - \gamma_i)t_2 + \gamma_i$ . Compute  $\tilde{f}_{\mathbf{c}} := \phi_{\mathbf{c}}(f) \in \mathbb{F}[x, t_1, t_2]$ , as a
   dense representation.
10  Factorize the trivariate  $\tilde{f}_{\mathbf{c}}$  over  $\mathbb{F}$ . Let  $S' = \{h_1(x, t_1, t_2), \dots, h_r(x, t_1, t_2)\}$  be the
   set of its irreducible factors.
11  for  $h_i \in S$  do
12    /* Computing the unique correspondence between bivariate and
   trivariate factors, and the evaluations */
13    Find the unique  $j \in [m]$  such that  $h_i(x, t, 0) = g_j(x, t)$ , if exists, otherwise go
   to the next factor in  $S$ .
14    Evaluate  $h_i(0, 0, 1)$ , and add  $(j, \mathbf{c}, h_i(0, 0, 1))$  to  $L'_j$ .
15  for  $j \in [m]$  do
16    /* Computing candidate sparse factors via interpolation and
   divisibility */
17    Use sparse interpolation algorithm (Lemma 13) to find an  $s$ -sparse polynomial
    $P_j$ , if exists, such that  $P_j(\mathbf{c}) = \theta$  where  $(j, \mathbf{c}, \theta) \in L'_j$ .
18    Check if  $P_j \mid f$ , or not. If yes, then update  $L' = \{P_j\} \cup L'$ .
19  return  $L'$ 
20 for each  $P \in L'$  do
21   /* Deciding irreducibility and computing multiplicities via
   Lemma 12 */
22   Check if  $P$  is irreducible, using Assumption 1 and its remark. If it is not
   irreducible, STOP, and go to the next polynomial in  $L'$ .
23   Otherwise, let  $z \in \{z_1, \dots, z_n\}$  be any variable that  $P$  depends on, so that
    $\partial_z(P) \neq 0$ .
24   Find the smallest  $e \geq 1$  such that  $P \nmid \frac{\partial^e f}{\partial z^e}$  and add  $(P, e)$  to the list  $L$ .
25 return  $L$ 

```

## ► Remarks.

1. Assumption 1 is true for constant degree polynomials and can be solved in polynomial time Corollary 16. Therefore, Algorithm 3 can be used to give an alternative proof of Theorem 18. This is because for  $f \in \mathcal{C}$ , we need to find  $\alpha$ , such that  $\text{Hom}_d[f](\alpha) \neq 0$ , which is given by the PIT oracle for  $\mathcal{C}$ . Once the  $\alpha$  is found,  $\mathcal{H}_\alpha$  can be found for the constant-degree polynomials, that preserves irreducibility; see Corollary 16. Additionally, the algorithm requires some divisibility testing by the candidate constant-degree polynomials, which is done using the divisibility testing  $\text{Div}(\mathcal{C}/\mathcal{D})$ .
2. Theorem 20 can be generalized to the input and output polynomials being computed by constant-depth circuits, by analogously changing the Assumption 1 for constant-depth circuits. In this case, we will only be able to output the factors as blackbox (because efficient reconstruction for constant-depth circuits is still unknown). Note that whether a constant-depth circuit divides another constant-depth circuit can be deterministically decided in subexponential time.

**6 Conclusion**

We conclude with some open questions.

1. Can we decide whether a given sparse polynomial is irreducible in deterministic subexponential time? The proof may already give a good bivariate projection that preserves irreducibility. Then Theorem 20 would give us a deterministic subexponential-time algorithm to find irreducible sparse factors of a sparse polynomial.
2. Can we find *bounded individual degree* sparse factors of a sparse polynomial (without any bound on the individual degree) in deterministic quasipolynomial time? Volkovich asked if multilinear factors of a sparse polynomial can be found in deterministic polynomial time [39].
3. Can one compute all the factors of a sparse polynomial/constant depth circuit by constant depth circuits of small size? At least, can one find all the factors that are computable in constant depth? The recent result in [29] gives a deterministic subexponential-time algorithm that outputs a list of circuits (of unbounded depth and possibly with division gates) that includes all such factors.
4. Given a blackbox computing the product of sparse irreducible polynomials  $f_i$  with bounded individual degree, find  $f_i$ 's in deterministic polynomial time. [3] gives a quasipolynomial time algorithm, when the input is sparse with constant individual degree and the factors are all sparse (polynomially upper bounded with respect to input polynomial's sparsity).

**References**

- 1 Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 3(23), 2003.
- 2 Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970.
- 3 Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. *Journal of the ACM (JACM)*, 67(2):1–28, 2020.
- 4 Pranav Bisht and Ilya Volkovich. On solving sparse polynomial factorization related problems. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*. Schloss-Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 5 Markus Bläser and Anurag Pandey. Polynomial identity testing for low degree polynomials with optimal randomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

- 6 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2005.
- 7 Nader Bshouty. Testers and their applications. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 327–352, 2014.
- 8 Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk. Optimal pseudorandom generators for low-degree polynomials over moderately large fields. *arXiv preprint*, 2024. [arXiv:2402.11915](https://arxiv.org/abs/2402.11915).
- 9 Michael A Forbes. Deterministic divisibility testing via shifted partial derivatives. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 451–465. IEEE, 2015.
- 10 Michael A Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 867–875, 2014.
- 11 Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.
- 12 Ankit Gupta. Algebraic geometric techniques for depth-4 pit & sylvester-gallai conjectures for varieties. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 21, 2014.
- 13 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *computational complexity*, 26:835–880, 2017.
- 14 Ming-Deh Huang and Yiu-Chung Wong. Extended hilbert irreducibility and its applications. *Journal of Algorithms*, 37(1):121–145, 2000.
- 15 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pages 355–364. ACM, 2003.
- 16 Erich Kaltofen. Computing with polynomials given by straight-line programs I: greatest common divisors. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 131–142, 1985.
- 17 Erich Kaltofen. Effective hilbert irreducibility. *Information and Control*, 66(3):123–137, 1985.
- 18 Erich Kaltofen. Polynomial-time reductions from multivariate to bi-and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469–489, 1985.
- 19 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989.
- 20 Erich Kaltofen. Effective noether irreducibility forms and applications. *Journal of Computer and System Sciences*, 50(2):274–295, 1995.
- 21 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. *Journal of Symbolic Computation*, 9(3):301–320, 1990.
- 22 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. 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 1409–1421. SIAM, 2011. [doi:10.1137/1.9781611973082.108](https://doi.org/10.1137/1.9781611973082.108).
- 23 Neeraj Kayal. Affine projections of polynomials. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 643–662, 2012.
- 24 Adam R Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 216–223, 2001.
- 25 Pascal Koiran and Nicolas Ressayre. Orbits of monomials and factorization into products of linear forms. *arXiv preprint*, 2018. [arXiv:1807.03663](https://arxiv.org/abs/1807.03663).
- 26 Pascal Koiran and Mateusz Skomra. Derandomization and absolute reconstruction for sums of powers of linear forms. *Theoretical Computer Science*, 887:63–84, 2021.

- 27 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015.
- 28 Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. Deterministic algorithms for low degree factors of constant depth circuits. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3901–3918. SIAM, 2024.
- 29 Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. Towards deterministic algorithms for constant-depth factors of constant-depth circuits. *arXiv preprint*, 2024. [arXiv:2403.01965](https://arxiv.org/abs/2403.01965).
- 30 Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.
- 31 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 804–814. IEEE, 2021.
- 32 Dori Medini and Amir Shpilka. Hitting sets and reconstruction for dense orbits in  $vp_{\{e\}}$  and  $\sigma\pi\sigma$  circuits. In *36th Computational Complexity Conference (CCC 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.
- 33 C Ramya and BV Raghavendra Rao. Linear projections of the vandermonde polynomial. *Theoretical Computer Science*, 795:165–182, 2019.
- 34 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *computational complexity*, 14:1–19, 2005.
- 35 Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I 35*, pages 60–71. Springer, 2008.
- 36 Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 284–293, 2007.
- 37 Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity (CCC 2016)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2016.
- 38 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.
- 39 Ilya Volkovich. Deterministically factoring sparse polynomials into multilinear factors and sums of univariate polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.
- 40 Ilya Volkovich. On some computations on sparse polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.
- 41 Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.